Active Learning in Recommender Systems: An Unbiased and Beyond-Accuracy Perspective

Diego Carraro
MSC
116224356

Thesis submitted for the degree of
Doctor of Philosophy

NATIONAL UNIVERSITY OF IRELAND, CORK

COLLEGE OF SCIENCE, ENGINEERING AND FOOD SCIENCE

SCHOOL OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY
INSIGHT CENTRE FOR DATA ANALYTICS

December 2020

Head of School: Professor Cormac Sreenan

Supervisors: Dr. Derek G. Bridge
Professor Barry O'Sullivan

Contents

List of Figures

List of Tables

I, Diego Carraro, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Diego Carraro

A Lorena, Fabio, Riccardo e Milú

Acknowledgements

I have learned and earned a lot since I came here in Cork in October 2016. The PhD is tough: it's about trying and failing and then trying again; then failing and then trying again and again. After a few iterations of such a loop, I have finally completed my PhD's dissertation. Some people deserve a special thank for this achievement and for these last four years in Ireland.

I would like to thank my supervisor Derek. His academic records speak for themselves, but I have known him personally, and I can tell more. His passion, dedication and work ethic are impressive. All these qualities have been inspiring to me. It was a pleasure to work under his guidance and to share also a little time outside our duties, where I found a very nice person. I feel lucky to had him on my side on this journey.

Andrea and Federico deserve special thanks. I have known them since the beginning of our computer science career in Treviso/Padova, and they are now family to me. They convinced me of embarking in the PhD, and for this, I will be grateful to them forever. Besides the lab, they are my companions on trips, nights out, sports and much more. And I am glad they will be with me in the next future too.

My colleagues at Insight have been part of this achievement. Francisco (Doctor Peña) and Mesut (Doctor Kaya), brilliant friends with whom I explored the Recommender Systems world, but also enjoyed barbecues and trips outside the lab. The Little Italy crew (with the adopted Cathal), with whom we imported the Italian style inside the lab (Scopetta above all).

Dedico mi trabajo a Judith, una persona muy especial con la que he compartido estos últimos meses difíciles de cuarentena y quien espero tener a mi lado mucho tiempo.

I am also grateful to Ireland and its people for the amazing times I have lived here in Cork. Impossible to mention everyone, but I dedicate my work especially to Enrico and Andrea, whose friendship has an important value to me.

Alle persone che ho "lasciato" in Italia. Questo traguardo è soprattutto merito della mia famiglia, senza la quale non sarei quello che sono. I miei genitori Fabio e Lorena e mio fratello Riccardo, che non mi hanno mai fatto mancare il loro affetto in ogni momento. Un pensiero per i miei nonni Renato e Giuseppe che sono venuti a mancare durante il mio periodo qui a Cork. E per le mie nonne Edda e Maria, che non smettono mai di chiedermi quando torneró a vivere in Italia.

Dedico il mio lavoro anche agli amici che non sono qui con me come vorrei, ma che non mancano mai di essere parte della mia vita. Alcuni meritano un ringraziamento speciale. A Soledad, un'amica imprescindibile. A Marco, importante come un fratello per me. Ad Angela, che ha sempre creduto in me. A Samuele, che completa il quartetto di ingegneri.

Finally, my thankfulness goes to all members and staff of the Insight Centre and the Computer Science department in UCC, especially to Professor Barry O'Sullivan. Their support and competence was priceless and made my journey much easier. This work has also been supported by a grant from Science Foundation Ireland (SFI) under Grant Number 12/RC/2289-P2, which is co-funded under the European Regional Development Fund.

Diego Carraro
Cork, December 2020

Abstract

The items that a Recommender System (RS) suggests to its users are typically ones that it thinks the user will like and want to consume. An RS that is good at its job is of interest not only to its customers but also to service providers, so they can secure long-term customers and increase revenue. Thus, there is a challenge in building better recommender systems.

One way to build a better RS is to improve the quality of the data on which the RS model is trained. An RS can use Active Learning (AL) to proactively acquire such data, with the goal of improving its model. The idea of AL for RS is to explicitly query the users, asking them to rate items which have not been rated yet. The items that a user will be asked to rate are known as the query items. Query items are different from recommendations. For example, the former may be items that the AL strategy predicts the user has already consumed, whereas the latter are ones that the RS predicts the user will like. In AL, query items are selected 'intelligently' by an *Active Learning strategy*. Different AL strategies take different approaches to identify the query items.

As with the evaluation of RSs, preliminary evaluation of AL strategies must be done offline. An offline evaluation can help to narrow the number of promising strategies that need to be evaluated in subsequent costly user trials and online experiments. Where the literature describes the offline evaluation of AL, the evaluation is typically quite narrow and incomplete: mostly, the focus is cold-start users; the impact of newly-acquired ratings on recommendation quality is usually measured only for those users who supplied those ratings; and impact is measured in terms of prediction accuracy or recommendation relevance. Furthermore, the traditional AL evaluation does not take into account the *bias problem*. As brought to light by recent RS literature, this is a problem that affects the offline evaluation of RS; it arises when a biased dataset is used to perform the evaluation. We argue that it is a problem that affects offline evaluation of AL strategies too.

The main focus of this dissertation is on the design and evaluation of AL strategies for RSs. We first design novel methods (designated WTD and WTD_H) that 'intervene' on a biased dataset to generate a new dataset with unbiased-like properties. Compared to the most similar approach proposed in the literature, we give empirical evidence, using two publicly-available datasets, that WTD and WTD_H are more effective at debiasing the evaluation of different

recommender system models.

We then propose a new framework for offline evaluation of AL for RS, which we believe facilitates a more authentic picture of the performances of the AL strategies under evaluation. In particular, our framework uses WTD or WTD_H to mitigate the bias, but it also assesses the impact of AL in a more comprehensive way than the traditional evaluation used in the literature. Our framework is more comprehensive in at least two ways. First, it segments users in more ways than is conventional and analyses the impact of AL on the different segments. Second, in the same way that RS evaluation has changed from a narrow focus on prediction accuracy and recommendation relevance to a wider consideration of so-called 'beyond-accuracy' criteria (such as diversity, serendipity and novelty), our framework extends the evaluation of AL strategies to also cover 'beyond-accuracy' criteria. Experimental results on two datasets show the effectiveness of our new framework.

Finally, we propose some new AL strategies of our own. In particular, our new AL strategies, instead of focusing exclusively on prediction accuracy and recommendation relevance, are designed to also enhance 'beyond-accuracy' criteria. We evaluate the new strategies using our more comprehensive evaluation framework.

Chapter 1

Introduction

1.1 Motivation and Background

In the digital age that we live in, it is increasingly difficult for a user to choose which products and services —henceforth, referred to collectively as 'items'— to consume. The items that a user is willing to consume are typically only the ones that she is interested in (those that are relevant to her), and they are a tiny fraction of the huge amount of the ones available. Recommender systems (RSs) are applications that help users to find such items. Applications such as e-commerce websites, social networks and streaming providers integrate recommenders into their platforms, to improve their services and increase revenue. For example, Amazon's recommender system proposes products to purchase; Netflix's recommender suggests movies to watch; and Spotify's recommender offers music playlists to listen to.

Given the importance of recommenders in the last couple of decades, the recommender systems research field has grown at an incredible pace. Probably, the most important key to a recommender system's success is the algorithm used to provide recommendations to users. Many approaches have been proposed in the literature and implemented in real-world applications, e.g. collaborative filtering algorithms, content-based algorithms, context-aware algorithms, among others [SG11].

Besides the algorithm, good recommenders rely on the quality of the user data to build their models. The more quality data they have available, the better the model. Such data can be collected by the RS in different ways: one way is to interview a user when she joins the system, typically asking for her demographic

details or for her opinions of a small subset of the items. Another way is to collect user data during the operation of the recommender system, when users interact with the items through the user interface of the system. Examples of this kind of data are a user's click on a particular item, or a user's rating or review after a product has been consumed by the same user.

Another option for an RS to collect explicit ratings data is Active Learning (AL). Essentially, Active Learning is about asking questions. In the literature, Active Learning was first applied and used mainly in classical machine learning problems, almost exclusively for supervised classification tasks [Set12]. In this scenario, Active Learning is when a machine learning model (usually called the 'learner') interactively queries some information source (usually called the 'oracle') to label new data points. The new labelled data points can be included in the training data and a new, hopefully improved model can be learned from the enlarged training set. Not all data points will be equally informative to the learner. Data points to be labelled are chosen 'intelligently' by means of an AL strategy. Different AL strategies take different approaches to identify the queries to submit to the oracle. For example, a widely-used approach is to query for those data points that the model is more uncertain about how to classify; another approach is to select those data points that will supposedly reduce the future error of the model.

This dissertation is about Active Learning applied to Recommender Systems. Oracles in the case of AL are the users of the system. The AL strategy selects some items and proactively solicits a user's opinions about them. To the best of our knowledge, all the research in this area assumes that these opinions will take the form of explicit feedback (usually numeric ratings for items), and this dissertation too makes the same assumption. In general, an AL strategy should select items that it believes will improve subsequent recommendation quality, but at the same time, they must also be items that the strategy believes will be familiar to the user, in order to get a successful response from the user.

AL strategies might be deployed during a user's regular use of a recommender. However, so far in the RS literature, AL has been almost exclusively employed to solve the cold-start problem [ME19]. This problem occurs, for example, during a user's sign-up phase, when the recommender has no data about the new user joining the system. This dissertation looks at the effect of using AL for more mature users too, by which we mean users who have more data in their profiles.

One crucial step when designing Active Learning strategies is their evaluation. AL strategies are typically evaluated offline first, but the traditional AL offline evaluation methodology described in the literature does not take into account the *bias problem*. This problem arises from the use of a *biased dataset* to perform the evaluation. The bias is a result of many factors, known as confounders [CSE18]. For example, the recommender's user interface is a confounder: differences in the ways items are exposed to users (e.g. position on the screen) influence the likelihood of a user interacting with those items. Some ways of mitigating the bias in the data used for the offline evaluation of RSs have been proposed in the literature, e.g. [SSS+16, BCC17]. However, to the best of our knowledge, no solutions for the bias problem have been proposed so far when evaluating AL approaches with offline experiments.

Besides the bias problem, in the offline evaluation of AL strategies, so far, the literature has exclusively focused on assessing the impact of AL on rating prediction accuracy and recommendation relevance. Nevertheless, nowadays, it is well recognized that a good recommender should provide recommendations that are not only relevant to a user, but, for example, also diverse, serendipitous and novel [KB17]. That is why we argue that an evaluation that does not account for the AL's impact in terms of such *beyond-accuracy* qualities is substantially incomplete. Moreover, we argue that designing new AL strategies that focus only on increasing prediction accuracy or recommendation relevance takes a view that is too narrow. An effective AL strategy should be designed to improve both accuracy and beyond-accuracy criteria.

In this dissertation, we fill some of the gaps in the literature that we have highlighted in this section. We propose a new evaluation methodology for Active Learning that mitigates the bias problem and offers a more comprehensive evaluation of an AL strategy. We use this tool to design new AL strategies that aim at improving accuracy and beyond-accuracy qualities. We also believe our framework opens up opportunities for practitioners to reconsider the effectiveness of strategies proposed in the literature, and to design new and more effective AL strategies to improve recommenders.

1.2 Contributions

The work in this dissertation is about Active Learning applied to Recommender Systems. In the following, we list our main contributions to this topic and how

they relate to the chapters of this dissertation.

A survey on the bias problem in the offline evaluation of RSs

To the best of our knowledge, we are the first to survey the bias problem in the offline evaluation of recommender systems, and the solutions that have been explored to cope with it (Chapter 2). In particular, we describe the use of unbiased datasets and unbiased metrics, as well as the generation of intervened datasets to debias the evaluation; and we discuss the pros and the cons for each of them. Additionally, we briefly review some works on mitigating the bias in training rather than evaluating a recommender. Some of this content is published in [CB20b].

WTD intervention method

We propose our own solution to debias the evaluation of an RS (Chapter 3). Our method, which we designate WTD (and its variant WTD_H), intervenes on biased data to generate data that is less biased. We compare WTD and WTD_H with SKEW [LCB16], the closest intervention approach to ours. For the first time in the literature, we provide empirical evidence that WTD, WTD_H and SKEW are valid methods to perform the desired debiasing action. With experimental results for two publicly-available datasets, we demonstrate that our solution more closely approximates the unbiased performances of different recommender algorithms and, additionally, it enjoys low overheads and high generality. Most of this content is published in [CB19] and [CB20b].

A comparison of AL in Machine Learning and Recommender Systems scenarios

To the best of our knowledge, we are the first to compare AL across the machine learning and recommender systems scenarios. We reveal similarities and differences in Active Learning's components, from both a theoretical and practical point of view (Chapter 4). In particular, we describe the characteristics of the oracles, the query, the strategy, the budget and the cost of AL, and how those relate to each other. Additionally, we complement the surveys of Settles [Set12] and Mehdi et al. [ERR16] in machine learning and recommender systems respectively, by proposing a high-level categorization of AL strategies that share the same principles in both scenarios. For each of the five main cate-

gories that we consider, we review some strategies that have been proposed in the literature.

A new, comprehensive evaluation framework for Active Learning

We are, to the best of our knowledge, the first to review the classical offline evaluation of Active Learning. Building upon this review, we propose a new offline evaluation framework for Active Learning, which we believe facilitates a truer picture of the performance of AL strategies (Chapter 5). The core feature that most distinguishes our more comprehensive evaluation from the narrow classic evaluation is that it mitigates the bias in the evaluation by means of the WTD or WTD_H intervention (a contribution presented in Chapter 3). Another distinguishing feature is that it can assess the impact of an AL strategy on *mature users* (by which we mean users who have more data in their profiles) as well as cold-start users, and users that are not queried by the AL strategy as well as users that are queried and especially those who are queried and who do provide new data. Our framework also assesses the impact of the AL on aspects of recommendation quality other than accuracy (such as diversity and serendipity). Using both a classic evaluation and one conducted using our new framework, we build up a case study that compares five simple AL strategies from the literature on two widely-used biased datasets. We find that the traditional biased evaluation shows different outcomes from our debiased one: this suggests that researchers should reconsider the effectiveness of their proposed strategies under these new findings. Most of these contributions are published in [CB18, CB20a].

AL strategies targeting beyond-accuracy objectives

We design new AL strategies and, differently from the ones that have been proposed so far in the literature, the focus of our strategies is on improving not only the accuracy of the system but also its beyond-accuracy qualities. We propose new strategies targeted at improving diversity, novelty and serendipity, and we do it by leveraging new tools that help us in such a design. Additionally, we investigate the use of hybridization approaches to AL again targeting beyond-accuracy criteria.

1.3 Publications

The following are publications produced from the work described in this dissertation:

- Diego Carraro and Derek Bridge: *A More Comprehensive Offline Evaluation of Active Learning in Recommender Systems*, Proceedings of the Workshop on Offline Evaluation for Recommender Systems (Workshop Programme of the 12th ACM Conference on Recommender Systems), 2018.
- Diego Carraro and Derek Bridge: *Debiased Offline Evaluation of Recommender Systems: A Weighted-Sampling Approach*, in Proceedings of REVEAL 2019, the Workshop on Reinforcement and Robust Estimators for Recommendation (Workshop Programme of the 13th ACM Conference on Recommender Systems), 2019.
- Diego Carraro and Derek Bridge: *Debiased Offline Evaluation of Recommender Systems: A Weighted-Sampling Approach*, in Proceedings of the 35th Annual ACM Symposium on Applied Computing (SAC '20), ACM, pp. 1435-1442.
- Diego Carraro and Derek Bridge: *Debiased Offline Evaluation of Active Learning in Recommender Systems*, in Proceedings of the 33rd International Florida Artificial Intelligence Research Society (FLAIRS) Conference, AAAI, pp.489-494, 2020.
- Diego Carraro and Derek Bridge: *Debiased Offline Evaluation of Recommender Systems: A Weighted-Sampling Approach*, in Proceedings of the Journal of Intelligent Information Systems (JIIS), submitted.

1.4 Outline of the Dissertation

The remainder of this dissertation is organised as follows.

In Chapter 2, we review the offline evaluation of recommender systems. After this review, we survey the bias problem in offline evaluation of RS and, briefly, in the training of a recommender.

In Chapter 3, we present WTD and WTD_H, our approaches to the debiased offline evaluation of RSs. We analyse the properties of biased and unbiased datasets first, and we use them to design our WTD and WTD_H intervention

methods. Then, through experiments on two datasets, we prove their effectiveness and compare them with SKEW, a state-of-the-art intervention method.

In Chapter 4, we provide an overview of the goals of Active Learning when applied to classical machine learning and recommender systems scenarios. We review the characteristics of the two. We compare them, and we review and categorise some AL strategies that have been proposed in the literature.

In Chapter 5, we review the traditional offline evaluation of Active Learning described in the literature. We propose a new offline evaluation framework for Active Learning, and we show its advantages by running a case study that evaluates five AL strategies from the literature.

In Chapter 6, we explore the use of Active Learning from a beyond-accuracy perspective. In particular, we propose new strategies that aim at improving diversity, novelty and serendipity, along with accuracy.

Chapter 7 draws conclusions and discusses open strands of research that might be explored in the future.

Chapter 2

Unbiased Offline Evaluation of Recommender Systems

The evaluation framework plays an important role when assessing the effectiveness of new recommendation approaches. Frameworks may rely on both offline and online settings. Indeed, the most common evaluation scenario includes an offline set of experiments as its first step, where the main goal is usually to measure and compare a set of candidate algorithms, or to tune the hyperparameters of a specific chosen model. Offline experiments are attractive because the costs of each (in terms, e.g., of design, implementation and run-time) is typically low. However, this comes, at the same time, at the expense of some compromises: one drawback is that designers must put in place some simplified conditions for the experiments, e.g. the behaviour of users interacting with the recommender must be simulated because real users are not available at this stage [SG11]. Thus, this first step is typically followed by more expensive user studies and online experiments where promising models can be further assessed in a more realistic setting. In A/B tests, for example, real users interact with different versions of a deployed recommender system.

In this dissertation, we focus on the offline evaluation of recommender systems and of active learning. In this chapter, we review the former. In particular, in Section 2.1 we describe a general framework for the classic evaluation of an RS. In Section 2.2, we investigate another of the drawbacks of the offline evaluation of RS, i.e. the widely-recognised bias problem. Then, in Sections 2.3, 2.4, 2.5, we review three different solutions to this problem that have been explored in the literature. To conclude, and for the sake of completeness, we review some

works on mitigating the bias in training rather than evaluating a recommender.

2.1 Offline Evaluation of Recommender Systems

We consider a recommendation scenario where we define a user-item space, $U \times I$, of size $|U| \cdot |I|$. We denote with $u \in U = \{1, .., |U|\}$ a generic user, and with $i \in I = \{1, .., |I|\}$ a generic item. Offline evaluation of a recommender system is done using an *observed dataset* D, which, within the user-item space, records interactions that occur between users and items during a given period in the operation of the recommender system. Without loss of generality, from now on, unless otherwise stated, we will consider such interactions to be numeric ratings. A different motivation for confining our presentation of this material to numeric ratings is the fact that this dissertation is about active learning, which involves soliciting explicit user feedback, also as numeric ratings (see Chapter 4).

We visualize D as a $|U| \times |I|$ matrix, i.e. $D \in (\mathbb{R} \cup \{\perp\})^{|U| \times |I|}$, where the $r_{u,i}$ entry records the rating given by the user u to the item i if the rating is observed, $\perp$ otherwise. We write $r_{u,i} \in D$ if the rating is observed, i.e. $r_{u,i} \neq \perp$, and $r_{u,i} \notin D$ if the rating is not observed, i.e. $r_{u,i} = \perp$. For RS, D is typically *sparse*, i.e. the number of observed ratings in D is much smaller than all the possible ratings in the whole user-item space. We will write $|D|$ for the number of real-valued entries, i.e. $|D| = |\{r_{u,i} \in D\}|$. Then, sparsity of the observed dataset means that $|D| << |U| \cdot |I|$. We denote with D_u the observed ratings of the user u and with D_i the observed ratings of the item i. We use I_{D_u} to indicate the set of items rated by the user u in D, i.e. $I_{D_u} = \{i \in I : r_{u,i} \in D_u\}$; we use U_{D_i} to indicate the set of users who rated the item i in D, i.e. $U_{D_i} = \{u \in U : r_{u,i} \in D_i\}$. We will also define the binary random variable $\mathcal{O} : U \times I \rightarrow \{0, 1\}$ over the set of user-item pairs in D as $\mathcal{O} = 1$ if the user-item rating is observed (real-valued) and $\mathcal{O} = 0$ otherwise (equal to $\perp$). (Later, however, when writing probabilities, we will use abbreviation $P(\mathcal{O})$ in place of $P(\mathcal{O} = 1)$.)

To evaluate an RS, the observed ratings in D are typically partitioned into a training set matrix D^{tr} and a test set matrix D^{te}; more generally, a training set and a test set are sampled from D. The training and test sets are typically obtained by performing a random split of the real-valued ratings in D but ignoring the values of the ratings. Other, more specific protocols are sometimes used. For example, we might split the ratings on a per-user basis, or we might leverage

side information such as the timestamp of the ratings to obtain a temporal split [Kor09, Lat10].

Once the data is prepared, the experiment consists of using the algorithm under evaluation to train a recommender model on the training set D^{tr}; then, the model is tested using the test set D^{te} to provide one or more measures of the quality of the algorithm under evaluation. Sometimes this process is performed k times, with k different training-test splits and results averaged across the k experiments.

As mentioned in Chapter 1, in early work in this field, there was a focus on accurate prediction of users' ratings. Hence, experiments used evaluation metrics that compare predicted and actual ratings for items in the test set D^{te}. Examples of such metrics are the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE). More recently, the focus has shifted to top-n recommendation, i.e. whether a recommender model correctly ranks the set of a user's unseen items and especially whether it correctly identifies and ranks the first n such items. For this, we use evaluation metrics such as Precision, Recall and Normalized Discounted Cumulative Gain (NDCG). They require a definition of what it means for a test item that is recommended during the experiment to be *relevant*. The typical definition is that a test item i is relevant to user u if $r_{u,i}$ exceeds some threshold. We write $\mathrm{rel}(u, i, D^{te})$ for the relevancy function such that $\mathrm{rel}(u, i, D^{te}) = 1$ if $r_{u,i}$ is in D^{te} and is a relevant item, i.e. exceeds a threshold, and is 0 otherwise. Specifically, for 1-5 star ratings datasets used in this dissertation, we define $\mathrm{rel}(u, i, D^{te}) = 1$ if $r_{u,i} \in D^{te} \wedge r_{u,i} > 3$ and 0 otherwise.

Let RL_u be the ranking of user u's unseen items, produced by a recommender model; we denote with $(\mathrm{RL}_u)_\mathrm{N}$ the first N elements of such a ranking. Finally, we denote with $\mathrm{r}(i, \mathrm{RL}_u)$ the position of item i in the ranking RL_u. The metrics we will use are defined as follows.

$$\mathrm{Precision@N} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{N}} \sum_{i \in (\mathrm{RL}_u)_\mathrm{N}} \mathrm{rel}(u, i, D^{te}) \tag{2.1}$$

$$\mathrm{Recall@N} = \frac{1}{|U|} \sum_{u \in U} \frac{\sum_{i \in (\mathrm{RL}_u)_\mathrm{N}} \mathrm{rel}(u, i, D^{te})}{\sum_{j \in D^{te}_u} \mathrm{rel}(u, j, D^{te})} \tag{2.2}$$

$$\mathrm{nDCG@N}_u = \frac{1}{|U|} \sum_{u \in U} \frac{\mathrm{DCG@N}_u}{\mathrm{IDCG@N}_u} \tag{2.3}$$

where

$$\mathrm{DCG@N}_u = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{IDCG@N}_u} \sum_{i \in (\mathrm{RL}_u)_\mathrm{N}} \left[\frac{\mathrm{rel}(u, i, D^{te})}{log_2(1 + \mathrm{r}(i, \mathrm{RL}_u))} \prod_{\substack{j \in \mathrm{RL}_u, \\ \mathrm{r}(j, \mathrm{RL}_u) < \mathrm{r}(i, \mathrm{RL}_u)}} (1 - \mathrm{rel}(u, j, D^{te})) \right]$$

and $\mathrm{IDCG@N}_u$ is the highest possible value of $\mathrm{nDCG@N}_u$.

In this dissertation, we use the phrase *accuracy metrics* when speaking about metrics that evaluate the accuracy of predictions of top-n recommendations, i.e. those mentioned above. However, during the testing phase, we can also use what we call *beyond-accuracy metrics*, to complement the accuracy metrics and give a more rounded evaluation of a recommender. In recent years, the focus of recommender systems research has shifted to include these beyond-accuracy qualities because the qualities they try to measure are recognized as crucial to fully satisfy and engage the users in real-life systems. A comprehensive review of beyond-accuracy qualities can be found, for example, in [KB17, CHV15]. In this dissertation, we consider diversity, novelty and serendipity. A diverse set of recommendations contains items that are different from one another, and therefore, it is more likely to contain one or more items that will satisfy the user. A novel set of recommendations contains items that are unknown to the user, i.e. unpopular among the users of the recommender system. A serendipitous set of recommendations contains items that are surprising, i.e. unexpected, to a user.

When measuring the beyond-accuracy performance of a recommender model, we need first to define a pairwise distance metric $\mathrm{dist}(i, j)$ between items i and j. In our case, the distance will be based on item features, such as movie genres; see Chapter 5 for the definition of $\mathrm{dist}(i, j)$ that we use for our datasets. We use Intra-List Distance, Popularity Complement and Content-Based Surprise beyond-accuracy metrics. We first define them in formulas 2.4, 2.7, 2.10 where they give a measure of diversity, novelty and serendipity on any set of items $J \subseteq I$, respectively. Then, for each, we derive two measures that can be applied to the recommendations lists provided by a recommender.

Intra-List Distance (ILD), first introduced in the recommender systems literature

by [SM01], is defined in Formula 2.4 as the average pairwise distance of the items in J. Then, Formula 2.5 is the average ILD value computed on the top-n recommendations of the users. Finally, using the definition provided in [VC11], we define the Expected Intra-List Distance (EILD) metric, also computed on the top-n recommendations of the users. Our version of EILD generalizes the ILD in Formula 2.5 by using the $\mathrm{rel}(.)$ function: in the resulting formula, the distance between pairs of recommended items is taken into account only as much as the two items are relevant for the target user u.

$$\mathrm{ILD}(J) = \frac{1}{|J|(|J|-1)} \sum_{i,j \in J} \mathrm{dist}(i,j) \tag{2.4}$$

$$\mathrm{ILD@N} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{N}(\mathrm{N}-1)} \sum_{i,j \in (\mathrm{RL}_u)_\mathrm{N}} \mathrm{dist}(i,j) \tag{2.5}$$

$$\mathrm{EILD@N} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{N}(\mathrm{N}-1)} \sum_{i,j \in (\mathrm{RL}_u)_\mathrm{N}} \mathrm{rel}(u,i,D^{te})\,\mathrm{rel}(u,j,D^{te})\,\mathrm{dist}(i,j) \tag{2.6}$$

Popularity Complement (PC) [VC11] is defined in Formula 2.7: it is based on the intuition that the more an item (or a recommended item) is novel, the more the item is unknown to all the users in the system (unpopular, e.g. less rated, interacted with, consumed). Formula 2.8 is the average PC value computed on the top-n recommendations of the users. Then, similarly to EILD, we define the Expected Popularity Complement (EPC) in Formula 2.9, where the novelty of a recommended item is taken into account only as much as the item is relevant for the target user u.

$$\mathrm{PC}(J) = \frac{1}{|J|} \sum_{i \in J} \left(1 - \frac{|D_i^{tr}|}{|U|}\right) \tag{2.7}$$

$$\mathrm{PC@N} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{N}} \sum_{i \in (\mathrm{RL}_u)_\mathrm{N}} \left(1 - \frac{|D_i^{tr}|}{|U|}\right) \tag{2.8}$$

$$\mathrm{EPC@N} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{N}} \sum_{i \in (\mathrm{RL}_u)_\mathrm{N}} \mathrm{rel}(u,i,D^{te}) \left(1 - \frac{|D_i^{tr}|}{|U|}\right) \tag{2.9}$$

In formulas 2.7, 2.8, 2.9, D_i^{tr} are the ratings of the item i available in D^{tr}, i.e. the popularity of the item i in the training set.

Content-Based Surprise (CBS) is defined in Formula 2.10: it is based on the intuition that an item (or a recommended item) is surprising if it is unlike any item the user has seen before. We use the lower-bound item distance from the items in the user's profile as an indicator of surprise, rather than the mean distance, because averaging the distance scores results in information loss (especially if the user has been exposed to diverse items) [KB14]. Formula 2.11 is the average CBS value computed on the top-n recommendations of the users. Then, similarly to EILD and EPC, we define the Expected Content-Based Surprise (ECBS) in Formula 2.12, where the serendipity of a recommended item is taken into account only as much as the item is relevant for the target user u.

$$\mathrm{CBS}(J) = \frac{1}{|J|} \sum_{i \in J} \min_{j \in I_u^{tr}} \mathrm{dist}(i, j) \tag{2.10}$$

$$\mathrm{CBS@N} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{N}} \sum_{i \in (\mathrm{RL}_u)_\mathrm{N}} \min_{j \in I_u^{tr}} \mathrm{dist}(i, j) \tag{2.11}$$

$$\mathrm{ECBS@N} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{\mathrm{N}} \sum_{i \in (\mathrm{RL}_u)_\mathrm{N}} \min_{j \in I_u^{tr}} \mathrm{rel}(u, i, D^{te}) \, \mathrm{dist}(i, j) \tag{2.12}$$

In formulas 2.10, 2.11, 2.12, $j \in I_u^{tr}$ is the set of items of the user u whose rating is in the training set D^{tr} and is conventionally referred to as the user's u profile.

2.2 The Bias Problem

It has been widely recognised in the literature that the observed datasets used in the offline evaluation of RSs are *biased*. The bias in a dataset is caused by many factors, known as confounders, that influenced the collection of the dataset ([CSE18, WLCB18]). For example, users usually experience what we can call *item discovery bias* because the RS acts as a confounder in the way that items are exposed to users [CnC18]. Indeed, the recommender's user-interface plays an important role as a confounder, e.g. the position of items on the screen influences the likelihood of a user interacting with those items [LCMB16]. Also, the recommender's algorithm sets up a feedback loop, which results in another confounder: users are typically more likely to interact with the recommender's suggestions than with other items. The user's preferences are also a confounder because they influence whether to consume an item or

not (*item consumption bias*) and whether to rate an item or not (*rating decision bias*). In a typical RS scenario, users are free to consume the item if they wish and, usually afterwards, they are free to rate the item or not. Their behaviour is often guided by their preferences on those items: for example, Marlin et al. demonstrate that, in a dataset of numeric ratings, the probability of not observing a specific user-item rating depends on the value associated with that particular rating: informally, users tend to rate items that they like [MZRS07]. User preferences and the characteristics of an RS are confounders that may also contribute to the so-called *item popularity bias*, i.e. the tendency of users to interact with popular or mainstream items rather than unpopular or niche items. This bias gives rise to the long-tail popularity curve [Har07], a well-known phenomenon in many RS datasets, where the distribution of the user interactions with items is skewed towards a few popular items [Cel08, ABM17]; see Figure 2.1 for an example. There are many publications that measure and explore popularity bias, for example, [PUG12] and [ABM17].

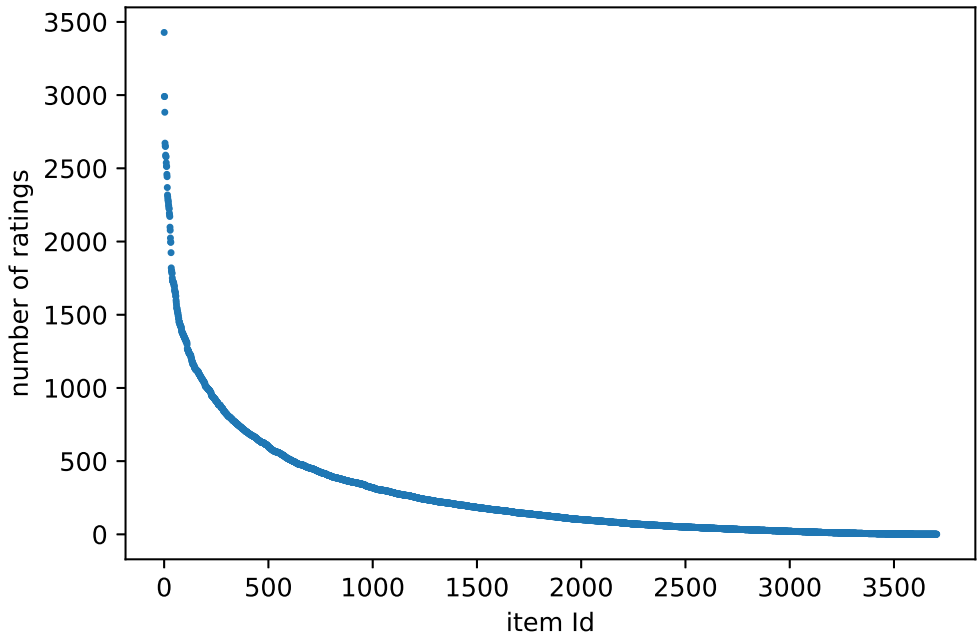

Figure 2.1: Long-tail popularity curve for Movielens 1M dataset. The items on the x-axis are ordered by decreasing number of ratings.

Because of these and other confounders, classical offline evaluations, which use biased observed datasets, result in biased (i.e. incorrect) estimates of a recommender's performance [MZRS07]. They are *biased evaluations*. For example, such experiments tend to incorrectly reward recommenders that recom-

mend popular items; or that make recommendations to the more active users; or that favour recommender approaches that exploit the bias in the dataset [PUG12, CKT10, BCC17].

Work in the RS field that seeks to handle the evaluation bias problem often leverages concepts from the fields of missing data analysis or causal inference. The missing data analysis theories, firstly proposed by Little and Rubin ([LR86]) and later introduced into the recommender systems literature by Marlin et al. ([MZRS07]), categorise different types of datasets based on so-called *missing data mechanisms*, which describe the process that generates the interaction patterns in the data. According to those theories, interactions that are missing from an observed dataset are Missing Not At Random (MNAR) [MZRS07] because of the many confounders, i.e. the dataset is biased. Nevertheless, classical offline evaluations using such an observed dataset are in effect making the assumption that missing interactions are either Missing Completely At Random (MCAR) or Missing At Random (MAR) instead [MZRS07]. (For the distinction between MCAR and MAR, see below.) Using MNAR data in an evaluation as if it were MCAR or MAR results in a biased evaluation.

In work on causal inference, the same missing data mechanisms are typically called the *assignment mechanisms* [IR15]. Roughly speaking, in a recommendation scenario the assignment mechanism exposes users to items and influences the interaction patterns of such users, e.g. analogously to exposing a patient to treatment and later observing its outcome in a medical study. As with the missing data mechanisms, ignoring the biased nature of the assignment mechanism (due to the presence of many confounders) most likely results in a biased evaluation.

In this dissertation, we use the missing data analysis terminology, i.e. MNAR, MAR, MCAR, and we want to make clearer how we use it in the following. In the literature, a distinction is sometimes drawn between Missing Completely At Random (MCAR) and Missing At Random (MAR). In [LR86, MZRS07], MCAR means that whether a user-item interaction is missing or not does not depend at all on interaction values (such as the values of the ratings in a recommender), i.e. it depends neither on the observed interaction values nor the missing interaction values. MAR, on the other hand, means that whether a user-item interaction is missing or not may depend on the observed interaction values, but is independent of the missing interaction values. However, in this dissertation, we use MNAR and MAR in a more informal and general way. We use

MNAR to indicate that data is biased because missing interactions depend on some confounders. Thus we use the terms MNAR and "biased" interchangeably. We use MAR to refer to data that is unbiased, where missing interactions do not depend on any confounder. Thus we use the terms MAR and "unbiased" interchangeably. Although these more informal usages are not properly in line with the categorisation in [LR86] and [MZRS07], our choice is broadly in line with other work in the recommender systems literature: what we refer to as MAR is also called MAR in papers such as [Ste10, CnC18]. But what we call MAR is also referred to as MCAR in papers such as [SSS+16], [KC14].

There are three ways by which experiment designers address the bias problem in the offline evaluation of RSs, and each of them will be reviewed in the following three sections. In Section 2.3, we describe a protocol to collect a MAR-like dataset which can be used instead of an MNAR dataset for the offline evaluation. In Sections 2.4 and 2.5, we describe unbiased metrics and debiasing interventions, respectively, two solutions that allow 'debiased' evaluations on MNAR data. There is also a substantial body of work that has been done in the last few years to cope with bias during the *training* of RSs models. For completeness, we review this work in the last section of this chapter (Section 2.6).

2.3 Collection of Unbiased Datasets

The 'straightforward' approach for coping with bias in the offline evaluation of an RS is to separately collect some unbiased data and use it as the test set. This can be done with what is sometimes called a "forced ratings approach" [CnC18]. For a ratings dataset, user-item pairs are chosen uniformly at random and for each user-item pair that gets selected the user is required ("forced") to provide a rating for the item. Thus, randomly-selected users are required to rate randomly-selected items. A dataset that is collected in this way will largely not exhibit the biases that we find in datasets that are collected during the normal operation of an RS (see Section 2.2). For example, we get rid of the item discovery bias and the popularity bias because items are randomly chosen: no confounders play any role in their selection. Items are not ones that are being exposed by the RS on the one hand, and users are not free to select them on the other hand [CnC18]; therefore, it is unlikely that we observe long-tail phenomenona in such datasets. The forced ratings approach also removes the item consumption bias because, unless the item was already known to the

user, users are forced to consume or interact with it so that they can rate it [CnC18]; in a typical RS scenario, by contrast, users are free to consume the item if they wish. The rating decision bias is also removed because users are not free to decide whether or not to rate the chosen item; they are required to rate it [CnC18].

However, datasets collected by the forced ratings approach are MAR-like, rather than MAR: they may still carry some bias. When building such a dataset, for example, although invitations are sent to users who are chosen uniformly at random, those who agree to participate may be atypical, thus introducing bias. Equally, the fact that, for each user, items to rate are presented sequentially introduces bias: the rating a user assigns to a particular item may be influenced by the items she has rated so far. Although this means that these datasets are less biased, rather than unbiased, to the best of our knowledge, this is still the best way of collecting this type of data.

Furthermore, the forced ratings approach can only work in certain domains; for example, it requires that a user who is presented with an item can quickly consume that item (or part of it) in order to form an opinion of it. In the movie domain, for example, we almost certainly cannot require a user to watch an entire movie (although we could require them to watch a movie trailer). Similarly, the forced ratings approach is impracticable in a tourism domain where a recommender suggests point-of-interests to its users: users cannot really be expected to visit the selected places in order to 'consume' and rate them (although we could require them to watch an advertisement video about such places).

Datasets collected by the forced ratings approach include Webscope R3 [MZRS07] and cm100k [CnC18] in the music domain, and CoatShopping [SSS+16] in the clothing domain. We will present these datasets in more detail and use them in Chapter 3.

2.4 Unbiased Metrics

The majority of the literature tries to overcome the bias in an MNAR test set by proposing new evaluation metrics which provide unbiased or nearly unbiased measures of performance on the MNAR test data. Such measures are supposed to be less affected by the bias in the data and therefore, more suitable for estimating the true performance of a recommender model. In some of the RS literature (e.g. [SSS+16, Ste10, Ste11, YCX+18]), metrics of this kind are

often called 'estimators': estimators are used in statistics to calculate an estimate of a given quantity of interest (a recommender's performance in our case) from observed data (test set data in our case). This terminology might suit an RS evaluation that takes a statistical framework perspective (e.g. [SSS+16]). However, in this dissertation, we prefer the term 'metric' to indicate a tool that measures a recommender performance based on available test data.

In [Ste10, Ste13], Steck designs ATOP, a new ranking metric that corrects for the biased measurements of Recall on an MNAR test set. However, this metric is unbiased only under two mild assumptions that must hold for the test set data. The first is that relevant ratings (which are typically a tiny fraction of all the possible ratings in the user-item space) are Missing At Random in the observed data. The second, regarding the non-relevant ratings, is that they are missing with a higher probability than the relevant ratings. In practice, the two assumptions allow the author to 'ignore' the missing data mechanism for non-relevant missing ratings (i.e. no missing data model is required). Also, there is no need for a missing data model for the missing relevant ratings at all (because they are missing at random). However, unbiasedness of ATOP is not always guaranteed, i.e. in datasets where these assumptions are unrealistic.

There is also work that tries to tackle specific biases in the data. For example, in [Ste11], Steck designs a modified version of the Recall metric that corrects for the long-tail item popularity bias. He modifies the definition of Recall by introducing weights that are proportional to the inverse popularity of the test set items. The resulting metric, which he calls Popularity-Stratified Recall, is considered a nearly unbiased metric under the assumption that no other confounders besides item popularity bias occur in the test data.

In [SSS+16], Schnabel et al. derive 'unbiased' versions of many widely-used metrics, both for ratings prediction (e.g. MAE and Mean Square Error) and top-n recommendation (e.g. Precision and nDCG). The 'unbiased' versions are based on the concept of Inverse-Propensity-Scoring (IPS) [IR15], [LR86], [Tho12]. A propensity score of a particular user-item pair $P_{u,i}$ is the probability of that pair being observed. IPS-based metrics use the propensity scores to weight the prediction/ranking errors on the test data computed by one of the standard metrics above. Schnabel et al. propose two different ways of estimating propensities for MNAR ratings datasets: one using Naive Bayes, and the other using Logistic Regression. While the former is an inexpensive approach, it requires a sample of MAR data; their approach to the latter does not require any additional MAR

data but it is instead more expensive and requires additional data (side data) about users and items (e.g. user gender and item features).

There is other work that uses IPS-based techniques to design unbiased metrics. For example, similarly to [SSS+16], Yang et al. in [YCX+18] propose new unbiased metrics to obtain widely-employed ranking measures (e.g. Recall and nDCG) on implicit MNAR datasets. The propensity score is modelled around the concept of item popularity and, in practice, it is calculated as the product of the probability that an item is recommended to the user and the probability that the user interacts with the item (given that the item has been recommended). However, the calculation makes strong assumptions about how data is generated. The assumptions include, for example, that the propensity scores are user-independent (i.e. $P_{u,i} = P_i$); that the user interacts with all the items she likes in the recommended set; and that her preferences are not affected by such recommendations. These assumptions do not hold in general, thus limiting the usefulness of this framework.

Lim et al. also propose a metric for implicit MNAR datasets [LML15]. They first assume a missing data model under which the observed dataset has been collected. Essentially, this model is that items that are relevant to users are Missing At Random. (This is like one of the assumptions in [Ste10]). Then, they design a novel evaluation measure, which they call Average Discounted Gain (ADG), that is built upon the nDCG metric. Unlike nDCG, they show that ADG allows unbiased estimation of top-n recommendation performances on test data which complies with their missing data model.

Finally, [KR20] is another interesting work on implicit datasets that investigates the effectiveness of what the authors call 'sampled metrics'. Sampled metrics are common evaluation metrics such as, for example, Precision and Recall, but used to measure a recommender's quality with a testing procedure that speeds up the evaluation (i.e. typically, the evaluation is performed by randomly sampling a small set of irrelevant items and ranking the relevant test set items only among this smaller set, instead of ranking test set items against the entire item catalogue), e.g. [CKT10, ESF18]. The authors show that a sampled metric can be a poor estimator of the true performances of recommender algorithms and suggest that the use of sampling in the evaluation should be avoided when possible. However, when sampling is required, the authors propose modifications that correct sampled metrics measurements and give a better estimate of the true performances (however, at the cost of increased variance in the results).

Although unbiased metrics, to some extent, achieve the desired goal of obtaining 'unbiased' measures of a recommender's performance, they suffer from some potential drawbacks. One of these is that they may not be general enough to overcome all sources of bias, i.e. they are often designed to compensate for a specific kind of bias (e.g. the popularity bias in [Ste11]). Another is that their unbiasedness might be proven only if the data used satisfies some specific conditions (e.g. the assumptions of [Ste10, Ste11, LML15] or the somewhat artificial recommendation scenario in [YCX+18]). Another drawback is that unbiased metrics might need additional data (e.g user gender and item features in [SSS+16]). Finally, they might require computationally expensive calculations (e.g. to estimate propensities in [SSS+16]).

2.5 Intervened Datasets

The third solution to the problem of bias uses what we will call an *intervention* approach, in contrast with the *regular* approach (with no intervention). In the latter, widely used in literature and explained earlier, the test set is typically generated by randomly sampling a portion of the available MNAR data, which gives rise to a biased RS evaluation. The former, instead, uses non-random sampling to produce a MAR-like test set, the *intervened test set*, which is supposedly less biased. The intervened test set is used in place of the regular (MNAR) test set to perform an unbiased RS evaluation.

The SKEW method by Liang et al. [LCB16] samples user-item pairs in inverse proportion to the item's popularity. This generates an intervened test set which has roughly uniform exposure distribution across items, thus reducing the item popularity bias in the test set. Liang et al. in [LCB16, WLCB18] and Bonner et al. in [BV18] use this technique for test set generation to evaluate causal approaches to recommendation. However, none of the three works that we have just cited either explain or verify empirically why SKEW should be effective as a debiasing technique. In this dissertation, we fill the gap by providing such contributions (see Chapter 3). Also, because of the similarity with our own work on debiased RS evaluation, we use SKEW as a state-of-the-art strategy to compare against our own approach (called WTD and WTD_H, see Chapter 3).

Cremonesi et al. in [CKT10] construct an intervened test set by removing ratings for the most popular items in the dataset from the MNAR test set, with the goal of mitigating the item popularity bias of the evaluation. In this way, a

recommender's quality is assessed on long-tail items only, while the recommendation of frequently-rated items is ignored. This is different from SKEW, which does not remove popular items but, rather, samples in inverse proportion to item popularity. Discarding all popular items may lead to specific insights but is generally too restrictive for a comprehensive evaluation. There is also a technical difficulty: given a specific dataset, it is not always clear what proportion of the items should be removed, leaving the evaluation quite arbitrary.

Bellogin et al. also sample an MNAR dataset to try to overcome the item popularity bias in the evaluation of a recommender, by means of two approaches [BCC17]. Their first approach (which is a percentile-based approach) is a form of stratification, in which training and test ratings are sampled from a partition of the data. In practice, the set of items is partitioned into m bins, based on item popularity, and the ratings of the items belonging to a bin form a popularity stratum. Then, for each stratum: a training set and a test set are sampled (typically by means of a random split of the ratings available in the stratum); and a recommender model is trained on the training set and tested on the test set. Results for the whole evaluation are obtained by averaging the recommender's performance across the m strata. One drawback of this methodology is the need to choose a value for the parameter m: it is not clear what m should be. The fact that the whole evaluation is broken down into m experiments is another drawback. The consequence is that an evaluation of this kind assesses to what extent a recommender is good at recommending items within a given popularity stratum. Bellogin et al.'s second approach (which they call Uniform Test Item Profiles) builds a test set with the same number of ratings for each item. However, this approach is very sensitive to the steepness of the item popularity curve. It may result in: generating quite small tests sets; and generating test sets where only a few popular items are included, therefore limiting the scope of the evaluation.

2.6 Unbiased Training of RSs

To conclude this review, and for completeness, we mention some of the work that has applied debiasing techniques for *training* recommender systems.

Bearing in mind that incorrect assumptions about missing data or assumptions that ignore the missing data mechanism may lead to biased inferences about users' tastes, some researchers decide to explicitly model the missing data mech-

anism. In [KC14, MZ09, HLHG14], for example, a probabilistic model of the missing data mechanism has been incorporated into existing (probabilistic) recommender algorithms to handle MNAR training data and improve learning.

Other works employ unbiased metrics as their loss functions when training their models and, therefore, to correct for the bias in the training set. For example, Steck uses an objective function based on (a surrogate measure of) the ATOP metric (presented earlier in Section 2.4) in a matrix factorization model [Ste10, Ste11]. Lim et al. design an efficient algorithm to optimize the ADG metric (presented earlier in Section 2.4) on MNAR implicit training data [LML15].

Another way of debiasing learning is to take a causal inference perspective, where a substantial body of work has been produced. In [JSS17], Joachims et al. propose a Propensity-Weighted Ranking SVM algorithm to train an unbiased search engine model on biased implicit feedback data. Schnabel et al. in [SSS+16] use propensities to derive a matrix factorization method for the task of ratings prediction on biased explicit datasets. In [LCB16, LCMB16], Liang et al. develop a causal inference approach which models the user exposure signals, i.e. how users discover items, along with user preferences, to correct for the exposure bias in the training set. Schnabel and Bennett implemented an algorithm for the item-to-item recommendation scenario based on causal inference, where propensities are estimated leveraging a small annotated dataset of hand-labeled data [SB20]. In [CTP+20], Christakopoulou et al. focus on the problem of estimating user satisfaction from user survey data. Given that such data is typically biased (i.e. a user tends to respond to a survey only when she is strongly satisfied with the item asked), to debias the training, authors employ an inverse propensity weighting technique to reweigh survey response examples by the inverse of their corresponding propensities to respond. Also Yuan et al. proposed a counterfactual framework, designed to unbiased the training of a Click-Through Rate (CTR) model for position-aware advertising systems [YLH+20].

Finally, there are publications that explicitly handle popularity bias in RS training. For example, in [ABM19], Abdollahpouri et al. explore the use of regularization in the objective function of a matrix factorization model to control the item popularity bias in the training set. In [PUG12], Pradel et al. employ the matrix factorization model of [Ste10] to debias the training. Their model uses a surrogate measure of the ATOP metric (see above) as the objective function; practically, the model is trained considering all possible ratings of the system,

regardless of whether a rating is observed or missing in the training set (in the latter case, a default rating value r_m is imputed). Also, during the training, ratings that are observed and are relevant are weighted differently with respect to ratings that are missing or that are observed but not relevant. Afterwards, the authors study the influence of different choices for such weights on the ranking performances and the popularity bias exhibited by the results.

In this chapter, we gave an overview of the offline evaluation of recommender systems, focusing on the bias problem which affects such evaluation. Then, we reviewed the main solutions proposed in the literature to solve this problem and, in the next chapter, we propose our own solution.

Chapter 3

Our Approach to Debiased Offline Evaluation of Recommender Systems

Designing an offline evaluation methodology which overcomes the bias problem in the data is crucial to obtaining reliable estimates of recommender performance. In Chapter 2, we have presented different solutions that can be found in the literature of the field. In this chapter, we explain and evaluate our own contribution to debiasing, which we call WTD (and its variant WTD_H). WTD and WTD_H are intervention methods (Section 2.5), where the intervention is performed on MNAR data before using it for the evaluation.

In this chapter, we first analyse properties of MAR and MNAR data (Sections 3.1 and 3.2). Subsequently, we use those properties to shape our WTD/ WTD_H intervention, a sampling strategy in which sampling weights are calculated by considering the divergence between the distribution of users and items in the MNAR data and their corresponding target MAR distributions (Section 3.3). Then, in Sections 3.4 and 3.5, we compare WTD and WTD_H with SKEW [LCB16], the closest intervention approach to ours. Our experiments on two different datasets allow us: to empirically evaluate for the first time the effectiveness of SKEW; to verify that SKEW, WTD and WTD_H successfully perform the desired debiasing action; but also to demonstrate that our strategy more closely approximates the unbiased performances of different recommender algorithms. Finally, in Section 3.6, we summarize the advantages of using WTD and WTD_H by discussing their properties.

3.1 Properties of a MAR Dataset

Using the notation presented in Section 2.1, we refer to two kinds of datasets over the same $U \times I$ space, $D^{mar} \in (\mathbb{R} \cup \{\perp\})^{|U| \times |I|}$ and $D^{mnar} \in (\mathbb{R} \cup \{\perp\})^{|U| \times |I|}$, which have MAR and MNAR properties respectively. In this section, we analyse properties of D^{mar} and in the next section the ones of D^{mnar}.

To generate D^{mar}, we make use of the forced ratings approach that we described in Section 2.3. First, we need to randomly sample a set of user-item pairs. Then, a preference for each pair is collected so that D^{mar} is obtained. (without loss of generality, we consider such a preference to be a numeric rating.) Note that, in order to satisfy the MAR property, the generation of D^{mar} is totally independent from the interaction values collected and from the particular identity of the user-item pair (u, i) as well. We also assume that, once the set of user-item pairs is determined, we can obtain the interaction values for all such pairs. (In practice, of course, users may decline the invitation to participate or may refuse to give some ratings, which is one reason why in reality these datasets are MAR-like and not MAR.)

Practically, to sample the set of user-items pairs, we make use of the probability distribution $P_{mar}(\mathcal{O}|u, i)$, defined over the space $U \times I$, that leads to D^{mar}. (We recall that we use the binary random variable $\mathcal{O}$ to indicate whether a rating is observed or not, see Section 2.1). A straightforward choice is to set $P_{mar}(\mathcal{O}|u, i) = P(\mathcal{O}) = \rho_{mar}$, where ρ_{mar} represents the desired ratio of observed entries from $U \times I$.

Now, assuming that a dataset D^{mar} has been collected using such an approach, we should empirically verify that user and item posterior probabilities are (roughly) uniformly distributed:

$$P_{mar}(u|\mathcal{O}) = \frac{|D_u^{mar}|}{|D^{mar}|} \approx \frac{1}{|U|} \quad \forall u \in U \tag{3.1}$$

$$P_{mar}(i|\mathcal{O}) = \frac{|D_i^{mar}|}{|D^{mar}|} \approx \frac{1}{|I|} \qquad \forall i \in I \tag{3.2}$$

where D_u^{mar} and D_i^{mar} are the observed ratings for user u and item i respectively.

Also, because users and items are drawn independently, we have that their posteriors are independent and we can write:

$$P_{mar}(u,i|\mathcal{O}) = P_{mar}(u|\mathcal{O})P_{mar}(i|\mathcal{O}) \approx \frac{1}{|U||I|} \quad \forall (u,i) \in U \times I \tag{3.3}$$

for the joint posterior of a specific user-item pair.

3.2 Properties of an MNAR Dataset

MNAR data is, of course, usually collected during the operation of a recommender system. But, similarly to the way we modelled the generation of MAR data D^{mar}, we can model the generation of an MNAR dataset D^{mnar} in terms of a drawing process.

Differently from the MAR scenario, due to the presence of bias, we cannot assume the sampling distribution P_{mnar} to be independent from the rating values D^{mnar} (or from other confounders too, including, e.g., the specific user-item pair (u,i)). In other words, in an MNAR dataset the draw is generally guided by some unknown probability $P_{mnar}(\mathcal{O}|u,i,Y,\mathcal{X})$, where Y represents the complete set of user-item ratings and $\mathcal{X}$ represents a set of features (covariates, confounders) which influences the sampling probability (e.g. user demographics, item features, characteristics of the system such as the way it exposes items to users, and so on).

If an MNAR dataset D^{mnar} has been collected, we can examine its user and item posterior probabilities, as we did for the MAR dataset but now, in general, we will find:

$$P_{mnar}(u|\mathcal{O}) = \frac{|D_u^{mnar}|}{|D^{mnar}|} \neq \frac{1}{|U|} \quad \forall u \in U \tag{3.4}$$

$$P_{mnar}(i|\mathcal{O}) = \frac{|D_i^{mnar}|}{|D^{mnar}|} \neq \frac{1}{|I|} \quad \forall i \in I \tag{3.5}$$

In general, the users and items are not uniformly distributed and thus, given that a specific entry is observed, i.e. $\mathcal{O} = 1$, we cannot assume user and item posterior independence for the joint posterior $P_{mnar}(u,i|\mathcal{O})$, i.e.

$$P_{mnar}(u,i|\mathcal{O}) \neq P_{mnar}(u|\mathcal{O})P_{mnar}(i|\mathcal{O}) \quad \forall (u,i) \in U \times I \tag{3.6}$$

However, the formulation that we have given here provides us with a solid framework to design our debiasing strategy in the next section.

3.3 Intervened Test Sets

To conduct unbiased evaluation from biased data, we generate and use intervened test sets in place of classical random heldout test sets. We begin by presenting this approach in general (Section 3.3.1), and then we present the specifics of our approach (Sections 3.3.2 and 3.3.3).

3.3.1 The intervention approach

The intervention approach consists in performing a debiasing intervention on MNAR data D^{mnar} by means of a given sampling strategy, denoted with S. The result of the intervention is the dataset D^S such that $|D^S| \leq |D^{mnar}|$ and with the objective that D^S has unbiased-like properties. Formally, we denote with $\mathcal{S} : U \times I \rightarrow \{0, 1\}$ the binary random variable that takes the value 1 when a particular user-item pair is sampled from D^{mnar}, 0 otherwise. (Again, we will use abbreviation $P(\mathcal{S})$ in place of $P(\mathcal{S} = 1)$.) A particular strategy S is characterized by the expression of the probability $P_S(\mathcal{S}|u, i, \mathcal{O}), \forall (u, i) \in D^{mnar}$, which is the probability distribution responsible for guiding the sampling on D^{mnar}. (In practice, only user-item pairs where a real-valued rating is available in D^{mnar} can be sampled.) We present our sampling approach in the next section.

3.3.2 WTD: weights for the sampling

In this section we present WTD, our debiasing intervention on MNAR data: we will start by assuming the availability of some MAR-like data D^{mar} in addition to MNAR data D^{mnar}. In fact, we will see in Section 3.3.3 that we can use our approach even in cases where we do not have any MAR data.

Our main idea is to make the posterior probability distribution of each user-item pair in the sampled D^S, i.e. $P_S(u, i|\mathcal{S})$, approximately the same as the posterior probability distribution observed for the corresponding user-item pair in D^{mar}, i.e. $P_{mar}(u, i|\mathcal{O})$. In other words, we want to make D^S similar to D^{mar} in terms of its posteriors. Writing this as a formula, we want:

$$P_S(u, i|\mathcal{S}) \approx P_{mar}(u, i|\mathcal{O}) \quad \forall (u, i) \in D^S \tag{3.7}$$

To obtain this approximation, we adjust the posterior distributions of the sampling space D^{mnar}, i.e. $P_{mnar}(u, i|\mathcal{O})$, using user-item weights $w = (w_{u,i})_{u \in U, i \in I}$

(similarly to [MGGR14]). We denote the modified weighted MNAR posteriors by $P_{mnar}(u, i|\mathcal{O}, w)$. The goal is to find weights w so that:

$$P_{mnar}(u, i|\mathcal{O}, w) = P_{mar}(u, i|\mathcal{O}) \quad \forall (u, i) \in D^{mnar} \tag{3.8}$$

From the fact that a typical MAR dataset is uniformly distributed over users and items, we use the independence of Formula 3.3 to re-write the right-hand side of Formula 3.8 to obtain:

$$P_{mnar}(u, i|\mathcal{O}, w) = P_{mar}(i|\mathcal{O})P_{mar}(u|\mathcal{O}) \quad \forall (u, i) \in D^{mnar} \tag{3.9}$$

Similarly to Formula 3.6, which considers user and item MNAR posteriors, user and item *weighted* MNAR posteriors will not in general be independent. However, we are going to treat them as if they were independent, to obtain the following:

$$P_{mnar}(u, i|\mathcal{O}, w) = P_{mnar}(i|\mathcal{O}, w)P_{mnar}(u|\mathcal{O}, w) \quad \forall (u, i) \in D^{mnar} \tag{3.10}$$

While Formula 3.10 is not true in general, we justify it by showing empirically in Section 3.5 that it does obtain good results.

Now, using 3.10, we can split Formula 3.9 into the two following equations:

$$P_{mnar}(u|\mathcal{O}, w) = P_{mar}(u|\mathcal{O}) \quad \forall u \in U \tag{3.11}$$

$$P_{mnar}(i|\mathcal{O}, w) = P_{mar}(i|\mathcal{O}) \quad \forall i \in I \tag{3.12}$$

As a consequence of formulas 3.11 and 3.12 for the weighted MNAR posteriors, we can define and calculate user-specific weights $w = (w_u)_{u \in U}$ and item-specific weights $w = (w_i)_{i \in I}$ instead of weights that are user-item specific. Having independent user and item weights also has an advantage in terms of scalability. We need to calculate only $|U| + |I|$ weights instead of $|U \times I|$. This is good for scalability because $|U \times I| >> |U| + |I|$ for the values of $|U|$ and $|I|$ that we typically find in recommender domains.

We propose the most straightforward solution to model the weighted MNAR posteriors, i.e. $P_{mnar}(.|\mathcal{O}, w) = w_. P_{mnar}(.|\mathcal{O})$. We plug this into formulas 3.11

and 3.12 and we obtain $w_u P_{mnar}(u|\mathcal{O}) = P_{mar}(u|\mathcal{O})$, $w_i P_{mnar}(i|\mathcal{O}) = P_{mar}(i|\mathcal{O})$ for each user and item weighted distribution respectively. Simply reversing these last two formulas, we have the expressions for calculating the weights:

$$w_u = \frac{P_{mar}(u|\mathcal{O})}{P_{mnar}(u|\mathcal{O})} \quad \forall u \in U \tag{3.13}$$

$$w_i = \frac{P_{mar}(i|\mathcal{O})}{P_{mnar}(i|\mathcal{O})} \quad \forall i \in I \tag{3.14}$$

We can think of the calculated weights as quantities that measure the divergence between the MNAR distributions of the sampling space and the target MAR distribution. Because a specific weight adjusts the corresponding MNAR distribution, we directly use weights to model the sampling distribution, i.e. $P_S(\mathcal{S}|u,i) = w_u w_i$. During the sampling, the effect of the weights is to increase or decrease the probability that a particular user-item pair is sampled depending on how divergent are the user and item posterior probabilities in the MNAR sampling space with respect to the MAR distributions.

In fact, based on preliminary experiments, we use $P_S(\mathcal{S}|u,i) = w_u(w_i)^2$ instead. This variant, denoted by WTD in the rest of this dissertation, raises the importance of the item-weight relative to the user weight. Specifically, $(w_i)^2$ will be bigger than w_i if w_i is greater than one, and $(w_i)^2$ will be smaller than w_i if w_i is less than one. This choice makes sense in the light of previous research reported in the literature which identifies item popularity as one of the most impactful confounders in MNAR data, e.g. [PUG12, Ste11].

3.3.3 WTD_H: hypothesized distributions for the weights

Up to this point, we assumed the availability of some MAR-like data in order to give us the posteriors that we need to approximate. But MAR-like data is expensive or impossible to collect, as we discussed when presenting the "forced ratings approach" earlier. Furthermore, in those cases where we do have a reasonable amount of MAR-like data at hand, we could use it directly as an unbiased test set. Using it to calculate weights so that we can intervene on MNAR data to produce a more MAR-like test set would then be pointless.

In fact, when we do not have any MAR-like data, we can still use our approach. We know that the posterior probability distribution for MAR data is uniform ($P_{mar}(u|\mathcal{O}) = 1/|U|$, $P_{mar}(i|\mathcal{O}) = 1/|I|$), and this is all we need for our sampling

approach. Therefore, we can use this hypothesized distribution when calculating the weights, avoiding the need for a MAR-like dataset. We call this strategy, WTD_H (where the H stands for "hypothesized").

3.4 Experiments

The goal of the offline experiments presented in this section is to assess the 'goodness' of different ways of producing intervened test sets. The measure of 'goodness' is how much results obtained by evaluating a recommender on an intervened test set resemble the results we would obtain on an unbiased test set. We assess our solutions, i.e. WTD and WTD_H, and compare them to SKEW [LCB16] and to two baselines, FULL and REG. We consider SKEW, which we presented in Section 2.5, to be the state-of-the-art strategy that most closely relates to our approach; FULL and REG perform a non intervention and a random intervention (which, in practice, is equivalent to no intervention) on MNAR data, respectively.

When deciding which intervention strategies to include in our investigation, we discarded some of the ones described in Section 2.5. Cremonesi et al.'s approach [CKT10] is one of them because it generates a test set devoid of ratings on the most popular items: it turns out that, by doing this, it is impossible to assess the quality of a recommender when recommending popular items, thus limiting the evaluation. We also do not include the two strategies of Bellogin et al., i.e. the percentile-based approach and the Uniform Test Item Profiles approach [BCC17]. The percentile-based approach trains a recommender and tests its performance on separate popularity segments of the item catalogue. Even though the quality of a recommender is inferred by averaging the performances across the segments, we argue that this approach still carries a similar limitation to Cremonesi et al.'s one (i.e. it compromises the representativeness of the whole experiment). The Uniform Test Item Profiles method also most likely discards ratings of some items (the least popular ones this time); and it may result in quite small test sets if the long-tail curve is very steep.

To note, the experiments in this section are published in [CB20b]. However, in this section, we report a more comprehensive analysis of the results (see Section 3.4.2 for the details).

Table 3.1: Datasets statistics

	WBR3		COAT	
	MAR	MNAR	MAR	MNAR
# ratings	54k	129k	4640	6960
# users	5400	5400	290	290
# items	1000	1000	300	300
avg. # ratings per user	10	23	16	24
avg. # ratings per item	54	129	15	23
avg. rating value	1.81	2.87	2.22	2.61
sparsity	0.99	0.97	0.94	0.92

3.4.1 Datasets

We use two publicly available ratings datasets: Webscope R3[1] (WBR3) from the music domain [MZRS07] and CoatShopping[2] (COAT) from the clothing domain [SSS+16]. Both of them are ideal for our purposes because they are composed of two parts, one having MAR properties (D^{mar}), and the other having MNAR properties (D^{mnar}). However, the two datasets have been collected in quite different recommender scenarios which, we argue, might influence our experimental results (see Section 3.5). Note that we did mention earlier (Section 2.5) that we know of one other MAR-like dataset, collected by the forced ratings approach, namely cm100k from the music domain [CnC18], but we cannot use this in our experiments because it does not have any corresponding MNAR data.

COAT's users are Amazon Mechanical Turkers who were asked (through a simple web-shop interface with facets and paging) firstly to find the coat they would have liked to buy the most and, afterwards, to freely rate 24 coats among the ones they had explored; those are the ratings that compose the D^{mnar} portion of the dataset. It is not clear for how long users were allowed to interact with the system. The forced ratings approach described earlier was used to additionally collect the D^{mar} portion of the dataset.

For WBR3, data was collected over a 20 days window. During this period, users used the LaunchCast Radio player, which gave them the freedom to rate songs (at any time and in any quantity) and receive personalised recommendations, and this produced the D^{mnar} portion of the dataset. Again, additionally the D^{mar} portion was collected using the forced ratings approach. It follows, for the reasons we gave earlier (see Section 2.5), that the D^{mar} portions of both WBR3 and COAT are almost but not completely unbiased.

[1] Available on request from `https://webscope.sandbox.yahoo.com`
[2] Available from `https://www.cs.cornell.edu/~schnabts/mnar/`

For both datasets, ratings are on a 1 to 5 scale; we also recall that we consider an item as relevant to a user if the item has a rating above 3, non-relevant otherwise (see Section 2.1).

For each dataset, we applied a preprocessing step to ensure that both D^{mar} and D^{mnar} have a common user-item space $U \times I$: specifically, we keep those users and items that belong to the intersection of the two portions. Table 3.1 gives statistics of the final resulting datasets that we used in the experiments.

3.4.2 Methodology

In our experiments, we randomly split D^{mnar} in each dataset into a training set D^{tr} and a heldout set D^{he} with proportions 60%-40% respectively. Since the split is random, MNAR distributions are preserved. D^{he} is what one would use as a traditional test set. But, in our case, we use D^{he} as the sampling space: we sample it to obtain different intervened test sets D^S. For each sampling strategy (REG, SKEW, WTD, WTD_H, explained in Section 3.4.3), we generate 10 different intervened test sets, each of which is obtained by sampling a portion ρ^p from D^{he}. The parameter ρ^p takes all the values in $\{0.1, 0.2, .., 1\}$ and represents the size of D^S with respect to the size of D^{he} (e.g. $\rho^p = 0.5$ means that $|D^S| = 0.5|D^{he}|$). We can view ρ^p as the parameter that guides the strength of the debiasing action on D^{he}: the smaller is ρ^p, the smaller but more debiased is D^S; the bigger is ρ^p, the bigger and less debiased is D^S, i.e. because it is more similar to D^{he}. In Section 3.5, we will see the impact of different ρ^p values on the results. To note, in the experiment of [CB20b], we only report results for $\rho^p = 0.5$.

We also randomly split D^{mar} into three, i.e. D^w, D^{val} and D^{gt} with proportions 15%-15%-70% respectively. Since the split is random, MAR distributions are preserved. D^w is used to calculate the weights for WTD (see Section 3.4.3 for more details of the calculation). We use D^{val} as the validation set to optimize recommender system hyperparameter values (Section 3.4.4). (In reality, the ratings one would use to optimize hyperparameter values would either be a portion of D^{tr} or a portion of an intervened test set produced from D^{he}. We decided it was better in the experiments that we report here to minimise the effect of hyperparameter selection on our results. Hence, we selected hyperparameter values using 'unbiased' data, D^{val}.)

We use D^{gt} as an unbiased test set. In other words, the performance of a given

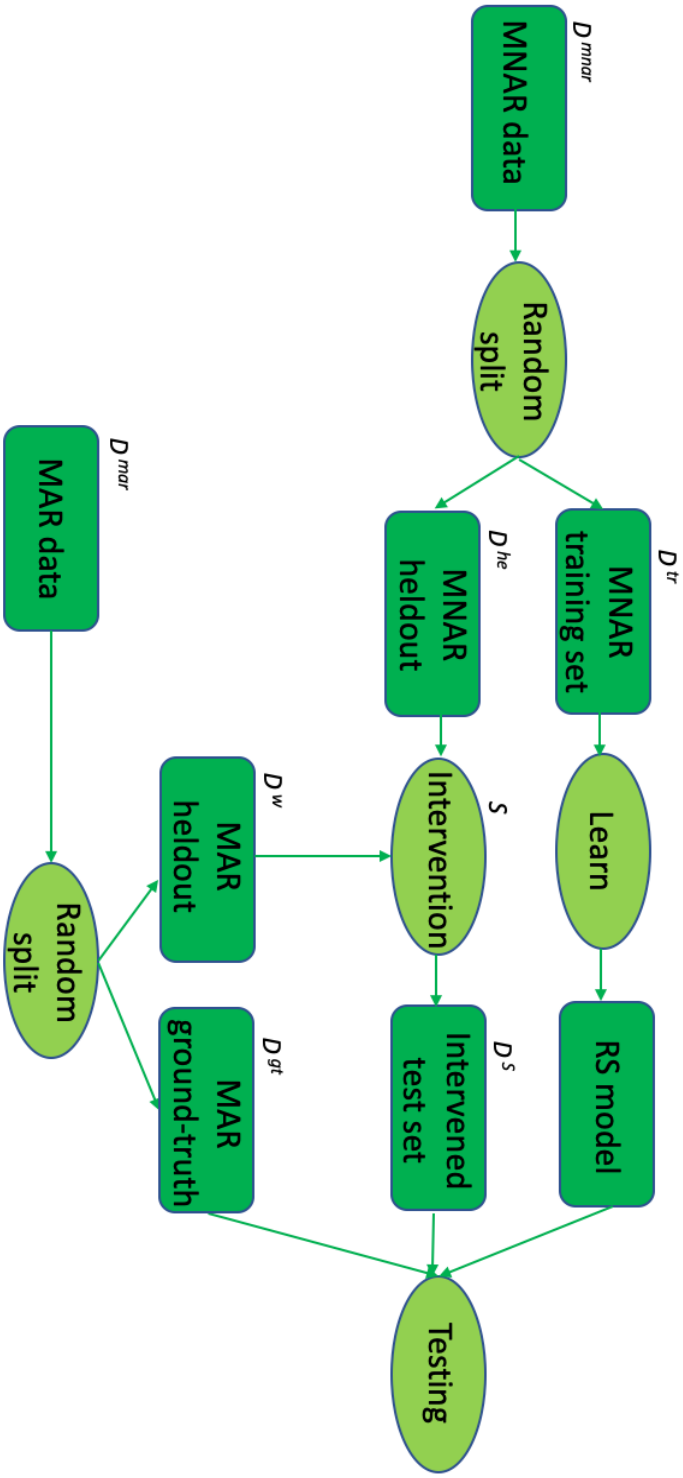

Figure 3.1: Visualization of the experimental methodology for each dataset.

recommender on D^{gt} can be considered to be its "true", unbiased performance (the ground-truth). We want the performance of a recommender on an intervened test set to be close to its performance on this unbiased test set. The best intervention strategy is the one that produces test sets where performance most closely resembles performance on D^{gt}.

We train the five recommender systems presented in Section 3.4.4 using ratings in D^{tr}. Each recommender produces a ranked list of recommendations which are tested on the unbiased test set D^{gt} and the intervened test sets. We have computed Precision, Recall, MAP and NDCG on the top-10 recommendations. Results are averaged over 10 runs with different random splits.

Figure 3.1 summarizes for each split the experimental methodology that we have just explained.

3.4.3 Sampling strategies for the intervention

We formally present here the sampling strategies that we use to produce the intervened test sets in our experiments. Each strategy samples an intervened test set D^S from D^{he}. For each strategy we give the corresponding probability sampling distribution, i.e. $P_S(\mathcal{S}|u,i)$. In addition to SKEW, WTD and WTD_H, we also employ two baselines. REG is a random sample from D^{he}, corresponding to an intervention that does not try to compensate for bias. FULL represents the test set in the classic evaluation, where the test set is D^{he} (therefore no intervention).

- **FULL**: $P_S(\mathcal{S}|u,i) = 1$. A test set sampled with FULL is what one would use as a traditional test set.
- **REG:** $P_S(\mathcal{S}|u,i) = 1/|D^{he}|$. Every (u,i) has a constant probability of being sampled and so we obtain a test set that is a random subset of D^{he}. We would expect this to behave very similarly to FULL test set, even though it is smaller.
- **SKEW:** $P_S(\mathcal{S}|u,i) = 1/|D_i^{tr}|$, where $|D_i^{tr}|$ counts the number of ratings that item i has in D^{tr} [WLCB18, BV18] (see also Section 2.5).
- **WTD**, **WTD_H**: $P_S(\mathcal{S}|u,i) = w_u(w_i)^2$. These are the two alternatives of our approach, presented in Sections 3.3.2 and 3.3.3. Weights are calculated using formulas 3.13 and 3.14. WTD uses formulas 3.1 and 3.2 to calculate the actual MAR posteriors from D^w. WTD_H uses the hypothesized MAR

posteriors instead. They both use formulas 3.4 and 3.5 to calculate exact MNAR posteriors from D^{tr}.

Note that, in each of SKEW, WTD and WTD_H, if the distribution P_S does not sum to 1 (necessary for a probability distribution), we include a normalization step on P_S to ensure that this property is achieved.

3.4.4 Recommender systems

We train five recommender models, all of them producing a ranked list of recommended items. AvgRating and PosPop are non-personalised recommenders which rank items in descending order of their mean rating and number of positive ratings in the training set, respectively. UB_KNN and IB_KNN are user-based and item-based nearest-neighbour algorithms [CKT10]. MF is the Matrix Factorization algorithm proposed by Pilaszy and Tikk [PZT10]. For UB_KNN, IB_KNN and MF we use the implementations available in the RankSys library[3]. We used our own implementations of AvgRating and PosPop.

The UB_KNN, IB_KNN and MF algorithms have hyperparameters. We select hyperparameter values that maximize Recall for top-10 recommendations on D^{val} (Section 3.4.2). For UB_KNN, IB_KNN, we choose the number of neighbors from $\{10, 20, .., 100\}$. For MF, we choose the number of latent factors from $\{20, 40, .., 200\}$ and the regularization term from $\{0.001, 0.006, 0.01, 0.06, 0.1, 0.6\}$.

3.5 Results

We report the results of our experiments in Figures 3.2 & 3.3 and Tables 3.2 & 3.3.

To analyse the difference between the various sampling strategies, we plot the distribution of the rating values of each of the intervened test sets and we compare them with the unbiased test set D^{gt} (similarly to Marlin et al.'s analysis in [MZRS07]).

Firstly, Figure 3.2 confirms the difference between unbiased (i.e. D^{gt}) and biased distributions (i.e. FULL and REG) for both datasets. In general, unbiased distributions show a much higher proportion of low ratings than high

[3] `https://github.com/RankSys`

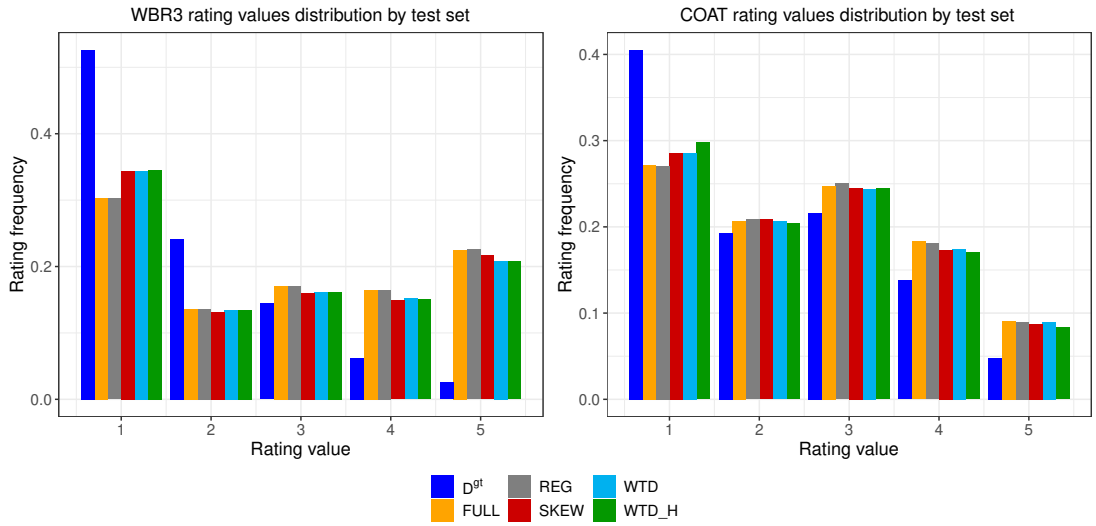

Figure 3.2: Distribution of rating values of the unbiased test set D^{gt}, the baselines and the intervened test sets in WBR3 and COAT.

ratings, confirming that in biased datasets users tend to rate items that they like [MZRS07]). This difference is less evident in COAT than WBR3 and we argue that this is due to the more artificial conditions under which COAT's MNAR portion was collected [SSS+16] compared with the MNAR portion of WBR3. WBR3's users experienced a standard recommender scenario (see Section 3.4.1) whereas COAT's users were not influenced by a recommender. The COAT users, being Mechanical Turkers, are mere executors of a task and therefore less likely to care about their experience of using the system; therefore, we argue that COAT is more randomized and accordingly less biased (i.e. more similar to an unbiased dataset). To confirm those findings, we observe values for FULL and REG in Table 3.2 where we report Kullback-Leibler (KL) divergence scores between the intervened sets and the ground truth for both datasets. This KL divergence is much greater for WBR3 (approximately 0.4) than it is for COAT (approximately 0.07).

Compared with FULL and REG, the distributions of rating values in the intervened test sets (i.e. SKEW, WTD and WTD_H) are closer to the distribution in the unbiased ground truth for both datasets (although only to a limited extent). This can be observed in both Figure 3.2 and Table 3.2, and it shows the first evidence that intervention might be a good solution to unbiased evaluation.

In Table 3.3, for each recommender, we show its ground-truth Recall@10 performance on the unbiased test set D^{gt} and its relative performance (in terms of percentage difference) on the baselines and intervened test sets with respect

Table 3.2: Kullback-Leibler (KL) divergence scores for WBR3 and COAT: scores represent the divergence of the baselines and the intervened test set rating values distribution with respect to the true unbiased rating values distribution of the unbiased test set D^{gt}.

	FULL	REG	SKEW	WTD	WTD_H
WBR3	0.417	0.417	0.359	0.347	0.343
COAT	0.072	0.073	0.058	0.06	0.047

Table 3.3: Recall@10 results for WBR3 and COAT. We report ground truth performances on test set D^{gt} in terms of Recall@10. We show the percentage difference of the best performances on the baselines and the intervened test sets with respect to D^{gt} (in brackets the test set proportion ρ^p where this best performance is achieved).

	D^{gt}	FULL	REG	SKEW	WTD	WTD_H
WBR3						
PosPop	0.056	+280	+244(0.1)	+6(0.6)	-14(0.7)	+32(0.8)
AvgRating	0.016	-77	-76(0.8)	-5(0.1)	-1(0.2)	-3(0.2)
UB_KNN	0.073	+274	+239(0.1)	+5(0.3)	-15(0.5)	+7(0.6)
IB_KNN	0.071	+313	+270(0.1)	-1(0.2)	-8(0.5)	+11(0.6)
MF	0.077	+258	+226(0.1)	+23(0.1)	-8(0.3)	-15(0.4)
COAT						
PosPop	0.066	+133	+108(0.2)	+17(0.1)	+7(0.6)	-1(0.7)
AvgRating	0.068	+61	+44(0.1)	+15(0.2)	-6(0.1)	+9(0.2)
UB_KNN	0.067	+229	+210(0.1)	+117(0.3)	+9(0.3)	+1(0.3)
IB_KNN	0.073	+236	+208(0.3)	+99(0.2)	+4(0.2)	-1(0.3)
MF	0.063	+180	+154(0.3)	+122(0.3)	+60(0.1)	+49(0.1)

to this ground-truth. For each of REG, SKEW, WTD and WTD_H, we show the best performance among the ones obtained in the 10 different test sets (one for each different ρ^p) and we show in brackets the test set size ρ^p for which this best performance is achieved. Results for Precision, NDCG and MAP are omitted because the percentage differences have a very similar trend to the Recall ones. The statistical significance of the results is assessed by performing a pairwise comparison test between the performance of each recommenders on the five different test sets, i.e. the baseline sets (FULL, REG) and the intervened sets (SKEW, WTD and WTD_H). For such tests, we use a two-tailed Wilcoxon signed rank test[4] with $p < 0.05$, and the results are reported in Table A.1.

Results on WBR3 show that WTD and WTD_H outperform SKEW only for the

[4]The version of the Wilcoxon test that we use includes zero-differences in the ranking process and assigns a zero rank to those differences. Also, it splits the zero ranks between positive and negative ones.

MF recommender (where all differences are statistically significant). This is however a good result if we consider that WTD and WTD_H are best at debiasing the evaluation of one of the most successful and widely-used recommenders in the literature [KBV09]. SKEW is superior to WTD and WTD_H for the PosPop and IB_KNN recommenders (with statistically significant differences). For the UB_KNN recommender, WTD_H and SKEW are equally good (their performances are not statistically significantly different) and superior to WTD; for the AvgRating recommender, all three are equally good because performances are not statistically significantly different from each other. The superiority of SKEW for PosPop is somehow expected because SKEW is an intervention that is specific to popularity-bias; its superiority for UB_KNN can be explained by a similar reason, i.e. UB_KNN has also been proved to be a recomender with a popularity-bias [CnC17].

We also observe that SKEW obtains its best performances on intervened sets that are smaller than the ones of WTD and WTD_H. However, this fact could raise questions about the reliability of SKEW's results due to discarding the majority of the available test data.

Comparing only WTD and WTD_H performances, we find that in general WTD is better than WTD_H, with the only exception being for the AvgRating recommender (where their performances are not statistically significantly different) and UB_KNN (where WTD_H is better than WTD).

The results for COAT in the lower half of Table 3.3 show that WTD and WTD_H are equally good because performances are not statistically significantly different from each other. Also, they more closely approximate the ground truth for the personalised recommenders but not for the non-personalised recommenders. Indeed, their performances are not statistically significantly different to the one of SKEW for PosPop and the ones of REG and SKEW for AvgRating.

Finally, in both datasets, baselines FULL and REG are very far from the ground-truth, showing that 'intelligent' intervention strategies provide an effective debiasing technique in offline evaluations. Indeed, SKEW, WTD, WTD_H achieve statistically significantly different performances with respect to FULL and REG with the exception of SKEW for MF on COAT. In general, FULL and REG have similar results, regardless of the fact that the best performances of REG is generally achieved on a test set which is much smaller than FULL (except for the one of AvgRating in WBR3). This means that what matters is the strategy that performs the sampling, rather than the sampling itself.

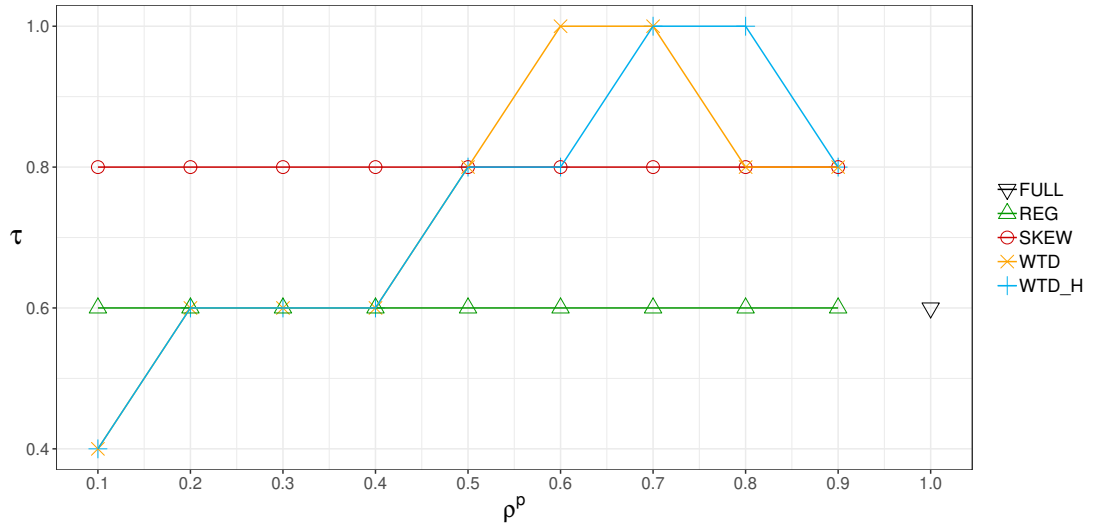

Figure 3.3: Kendall's concordance coefficient (τ) values for WBR3.

Figure 3.3 reports an additional investigation on the results of Table 3.3. An offline evaluation typically ranks recommender algorithms from best to worst. This helps to narrow the number of different recommender algorithms that needs to be evaluated in costly user trials and online experiments. In our case then, it is important that performance estimates on intervened test sets, not only get close to the ground truth performance, but also rank different recommenders in the same way they would be ranked by performance estimates on the unbiased test set.

Before seeing whether the ranking of the recommenders on intervened sets corresponds to their ranking on the ground truth, we wanted to make sure that the ground truth ranking was reliable. Thus, we first computed statistical significant tests on the ground truth ranking. The statistical significance of the results is assessed by performing a pairwise comparison test between the performances of the recommenders on the unbiased test set D^{gt}, again using the two-tailed Wilcoxon signed rank test described earlier (see Tables A.2, A.3). We found that, for WBR3, recommender performances are statistically significantly different from each other, except for the pair UB_KNN & IB_KNN. Unfortunately, for COAT, no recommender performance is statistically significantly different from any other, except for the pair MF & IB_KNN. We argue that this is due to the small size of the COAT training set. This means that for COAT there is no point in comparing the rankings produced by the different intervened test sets, because all recommenders are roughly equivalent according to the ground truth test set.

We use Kendall's concordance coefficient (τ) to compare the ground truth recommender ranking obtained on the unbiased test set with the ones produced by the different interventions. For the reasons above, Figure 3.3 reports the results for WBR3 only: for each of the intervention approaches we show concordance coefficients obtained in their 10 different intervened test sets. The figure shows that the 'intelligent' interventions are superior to FULL and REG, i.e. SKEW, WTD and WTD_H have values no smaller than the ones of REG (with the only exception of WTD & WTD_H when $\rho^p = 0.1$).

In more detail, FULL, REG and SKEW have constant τ values (0.6, 0.6 and 0.8, respectively), with SKEW being the best of the three. WTD and WTD_H have different values, depending on the size of their test sets. In general, both are superior to SKEW from $\rho^p = 0.9$ down to $\rho^p = 0.6$, achieving perfect correlation ($\tau = 1$) when $\rho^p = 0.8$ (WTD_H), $\rho^p = 0.7$ (WTD & WTD_H) and $\rho^p = 0.6$ (WTD). SKEW, WTD and WTD_H have $\tau = 0.8$ for $\rho^p = 0.6$, but SKEW is superior to all the other strategies from $\rho^p = 0.1$ up to $\rho^p = 0.4$ inclusive. We would argue that the results obtained by our debiasing strategies are more valuable than those of SKEW and REG because they are superior when sampling most of the data available for testing. Indeed, ρ^p values smaller than 0.5 can result in intervened test sets that are too small to give reliable results.

3.6 Conclusions

In this chapter, we presented WTD and WTD_H, our new sampling strategies that generate intervened test sets with MAR-like properties from MNAR data. These intervened test sets are more suitable for estimating how a recommender would perform on unbiased test data. One of the sampling strategies, WTD, requires that some MAR-like data be available since it approximates posterior probabilities calculated from that data. The other strategy, WTD_H, approximates the probabilities that we expect MAR data to exhibit.

The chapter assesses the effectiveness of these two strategies and it assesses, for the first time, the effectiveness of an existing intervention strategy from the literature, namely SKEW, which samples from MNAR data in inverse proportion to item popularity. With the use of essentially unbiased test sets as ground-truth, we showed these three sampling approaches to be successful in mitigating the biases found in a classical random test set. In general, we found SKEW to be particularly good at reducing the bias for well-known recommenders that them-

selves suffer from a popularity-bias (i.e. PosPop and both nearest-neighbour recommenders). Item popularity-bias is the kind of bias for which SKEW was designed. But our new strategies are the most robust across various recommenders (MF on WBR3 and all the personalized recommenders on COAT) since they most closely approximate the unbiased ground-truth performances. The WTD strategy requires MAR data, which is rarely available, but we found that WTD_H, which uses a hypothesized MAR distribution, does work well, so MAR data is not necessary.

Our approach brings several intrinsic benefits. First of all, it enjoys low overheads.

- Its design is simple and easy to implement and it does not require any learning phase for the weights, contrary to some unbiased estimators which might require expensive learning (e.g. [SSS+16], where propensities are found via logistic regression).
- Moreover, intervention reduces the computational costs of testing a recommender because it generates smaller test sets.

Another advantage of our approach is that it has high generality.

- It works for both implicit and explicit datasets because it is independent of the interaction values (e.g. ratings) in the dataset.
- Despite the fact that WTD and WTD_H are close to SKEW on some recommenders, our way of calculating weights is less heuristic than the one of SKEW and, unlike SKEW, it is not tailored to item popularity-bias.
- Our approach can be extended to training a recommender, without any modification. Training a recommender on an intervened training set, instead of on a classical biased training set, might improve the recommender's model and therefore boost prediction or top-n recommendation performances. We leave this as future work.
- Intervened data can be used to train *existing* recommender systems and to test recommender systems using *existing* metrics. Debiased training and testing hence become widely applicable without designing special models and special metrics. This feature is particularly desirable when benchmarking new recommender approaches with respect to existing ones.

Chapter 4

Active Learning and Recommender Systems

Active Learning (AL) is the sub-area of Machine Learning ML in which a learning algorithm seeks to improve its performance by asking questions. Even though those questions might be posed in different forms by the learner, most of the AL literature is related with the labelling process, where questions are posed to oracles and take the form of data points to be labelled so that they can be used in a supervised or semi-supervised classic ML setting (almost exclusively for classification tasks). Most recommender systems use interaction data, similar to the labels used in classic supervised learning, and are therefore amenable to a similar kind of active learning.

In Section 4.1, we describe the principles of AL, i.e. its goals, its components and how the latter interact with each other in the AL cycle. In Sections 4.2 and 4.3, we frame AL within ML and RS scenarios respectively: we compare the two fields by exploring similarities and differences from both a theoretical and practical point of view. Finally, in Section 4.4, we present some AL strategies that have been applied within ML and RS scenarios, framing each of these strategies into one of five main categories that we consider. Methodologies for evaluating AL strategies are also important, but we will postpone reviewing the appropriate literature on this topic until we make our own proposals in the next chapter.

4.1 The Active Learning Cycle

In supervised settings, learning algorithms make use of labelled data instances to train a model, such as a classifier or a recommender, which is afterwards able to perform new tasks (e.g. image classification, spam filtering, or movie recommendation). In order to perform adequately, quite often such a model needs to be trained on a considerable number of labelled instances. However, getting labels for instances might be expensive in terms of time, effort and cost, while, on the other hand, unlabelled data is often available in large volumes in our present data-driven society. One example (in ML) is the automatic document classification task, where unlabelled data (i.e. documents) is available on the web in large quantities, but it usually requires manual labelling from human annotators, which is a slow and challenging process. Another example (in RS) is an online retailer recommender, where the item catalogue might be composed of millions of products (e.g. as is the case with Amazon) but having enough user data (e.g. ratings or reviews) for each of those items is often challenging due to the low rate of user feedback.

For the above reasons, the labelling process might be a bottleneck in many supervised settings: reducing its cost can be achieved using Active Learning. Indeed, a strong motivation behind the use of AL is that not all the labels are informative in the same way, i.e. labelling some of those unlabelled instances would make little or no difference from the perspective of the learner. Obtaining the label for a data point might result in having noisy or duplicated training data, or data which has no useful features for the learning task. For example, for many kinds of models such as Support Vector Machines (SVM), labelling examples that do not lie on any class boundaries might make no difference to what is learned. AL is therefore applied under the following belief: *if it was the learner who directly chose data instances to be labelled, then it would select the most informative ones.* If this happens, on the one hand, the expense for new training data would be reduced because only useful data would be labelled and used for training. On the other hand, the learner's utility would be increased thanks to this new informative labelled data. It turns out that the ultimate goal of AL is subject to a *trade-off between utility and cost*, and the effectiveness of every AL approach depends on it.

We can divide an Active Learning framework into different components that interact with each other in what we can call the *Active Learning cycle*. In Figure 4.1, we depict the AL cycle in the typical supervised setting and in the following,

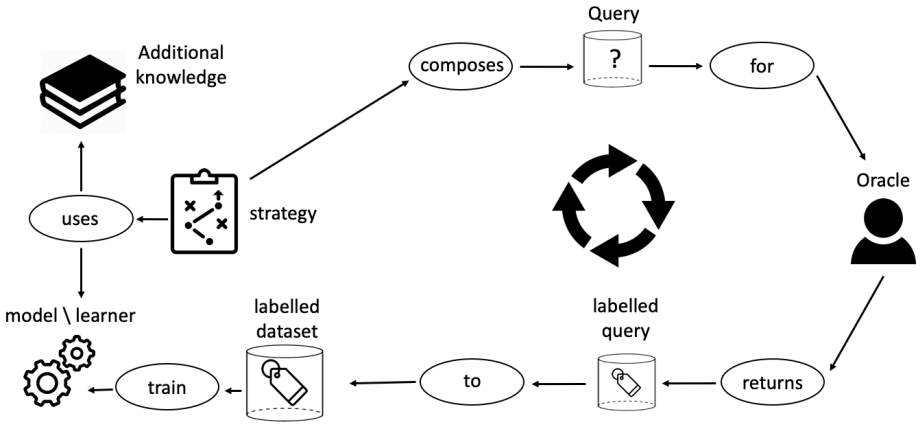

Figure 4.1: The AL cycle in the typical supervised setting.

we discuss its operation.

In the common AL terminology, the *learner* is responsible for building the *query* that is posed to the *oracle*; the oracle is the information source that is responsible for providing an answer to the query, which is used to update the model that is being learned. The relationship between the learner and the oracle can be seen as the one between a student (i.e. the learner) and a teacher (i.e. the oracle): in an AL framework, oracles are supposed to be experts in the domain of the application and therefore able to be reliable information sources; the learner instead proactively gathers new information from the oracle to improve its effectiveness at accomplishing a specific task in the domain of the application. However, in this dissertation, we partition the role performed by the learner into three components, i.e. the strategy, some additional sources of knowledge, and the learning algorithm with the model that it learns, to make more of the details of its operation explicit.

In more detail, the *strategy* is the component which builds the content of the query: this means the query should contain an information request that, if successful, will *maximize* utility to the learner while *minimizing* the associated acquisition cost to the learner and the oracle. To do so, the strategy often uses the characteristics of the model and perhaps some *additional knowledge sources*. Characteristics of the model might be, for example, the type of the model (e.g. for a classifier, SVM, Naive Bayes, etc.), or its actual 'beliefs' (e.g. the

support vectors of an SVM classifier); additional knowledge sources might be, for example, information about the nature of the oracles (e.g. whether they are human annotators, or whether they are other models) or a knowledge source might even be another model that can help select the content of the query.

The vast majority of AL approaches in the literature, both in ML and RS scenarios, investigate the use of *pool-based sampling*, where the strategy typically selects one unlabelled data point from a pool of available unlabelled data points, i.e. a *candidates set*. Concretely, the strategy evaluates the informativeness of all of the data points (one by one) in the candidates set and chooses the one to query. An alternative to pool-based sampling is *stream-based sampling*: in this case, firstly, one single unlabelled data point is sampled from the unlabelled input source available. Then, the strategy makes a real-time choice about to whether or not to select the data point for the query (discarding it otherwise). When discarding the data point, another data point is sampled and evaluated for the query, and so on. While pool-based sampling might require expensive computational times in assessing the informativeness of all of the data points available, stream-based sampling supposedly enjoys reduced computational time instead, because many fewer data points are evaluated. Examples of its application are [ACL+90, CAL94, Yu05, MNS+07a]. A third well-known approach in the literature is *query synthesis sampling*, where the strategy synthesizes (i.e. artificially creates) the data point which composes the query (and the synthesis usually leverages information from the data input space). However, labelling artificial data might be a problem if, for example, such generated data is ambiguous or difficult to interpret for the oracles (e.g. in image classification tasks, when images make no sense for human annotators). Query synthesis is used, for example, in [LB92, KWJ+04] and it is particularly suitable when there is a lack of unlabelled data for a particular task.

Once the query is submitted to the oracle, she may label the data point in the query and return its label. The new labelled data point is therefore available in addition to the already-labelled dataset and can be used to train a new and hopefully better model. Training the model after each new incoming label might be an expensive process. To reduce the expense, an alternative is to fill the query with more than one data point (sometimes called *batch mode Active Learning*), so that the model is only trained after a consistent amount of new data is labelled. Under this setting, stream-based and query synthesis samplings repeat their single-selection procedures until n data points are selected to compose the query for the oracle. In pool-based sampling, the strategy selects n

data points to compose the query, among the m available in the pool. In the following chapters of this dissertation, we will evaluate and design active learning approaches where the query is composed of more than one unlabelled data point. But, in the rest of this chapter that reviews the AL literature, unless said otherwise, we will consider the traditional single-instance query when referring to a query.

What we have just described is one iteration of the AL cycle. Typically, this cycle is repeated multiple times and stopped when some conditions are matched, e.g. a maximum number of iterations is reached, or a budget for querying is spent; see Sections 4.2 and 4.3.

So far, we described Active Learning in relation to a labelling process, i.e. the query is composed of unlabelled data and oracles are asked to label the data. However, variants of Active Learning can also, for example, be used to ask for rules, advice, or feature values; see Sections 4.2 and 4.3 for examples of this kind.

In the next two sections, we will explore more closely applications of AL on the two different scenarios of our interest: ML and RS.

4.2 Active Learning in Machine Learning Scenarios

As mentioned earlier, most of the literature explores the use of AL in classical ML scenarios, i.e. in supervised settings and mostly for classification tasks. For example, under the framework in Section 4.1, we can imagine building a fraud detection model that can detect fraudulent transactions in the activities within an organisation. In order to improve such a model, an AL approach is designed to query a single human, financial expert (i.e. the oracle, who can always provide the correct answer to the query) for new labels; the queries should contain unlabelled transactions that would increase the accuracy of the classifier while minimizing the associated acquisition cost of such labels. As typically happens, we assume the labelling cost is proportional to the number of unlabelled transactions queried, i.e. the fewer unlabelled transactions queried to the oracle, the lower the cost. Thus, a good AL strategy will improve the classifier at a low expense. However, such a scenario very often does not reflect the situation of real-world applications because it may make oversimplified assumptions

regarding many components of the AL framework. In the following sections, we plot some relaxations to this scenario and discuss AL approaches that are designed according to these alternative scenarios.

The cost and the budget of AL

An assumption in the example given previously regards the cost of AL, calculated in terms of the number of queries submitted to the oracle. However, the cost of querying can be expressed in terms of effort, money, additional data required, etc. A more realistic approach is calculating the cost as a function that depends on the strategy applied, the query and the oracles at the same time. Some AL strategies might require little or no computations to select the content for the query, while some might require expensive computations instead. Some queries might require more effort than others to be labelled: an example is in a text document annotation task, where long documents often require more time to label compared with short ones. Another example is related to the characteristics of the oracle: some oracles might demand higher pay than others; some might be faster than others in labelling. All this should be taken into account by the designers of the AL; but the dynamic nature of the AL cost is not easy to capture and to model and it is highly domain-dependent. That is why, as we said, most of the literature relies on the number of question asked to oracles or even avoids modelling the costs at all.

While most of the ML literature takes a simple view of the AL costs or avoids modelling them, there are a few exceptions. For example, cost-sensitive approaches have been proposed by [BO04, CM05a, CKMV06, FRH+12], where the authors use an automatic pre-annotation step, i.e. the learner's predictions are used to entirely or partially label queries before being submitted to the oracle. The goal of this approach is, therefore, to reduce the workload of the oracle, which should place its effort on what is left to label. However, in general, automatic pre-annotation does not model the intrinsic cost of labelling (e.g. it does not establish the cost associated with labelling a text document), but only tries to reduce it by indirectly minimizing the effort of the oracles.

In other work, such as [KWJ+04, KHB07], the AL strategy aims at selecting the cheapest query to be labelled, making its choice based on a query-based model cost. There is also work where the cost of labelling is subject to some dynamic domain conditions (e.g. the cost is a function of the elapsed annotation time) and can be predicted; examples are [SCF08, ANR09, HRC08, VG09].

An aspect of AL that is cost-related and can be taken into account in AL design is the AL budget, i.e. the amount of resource available to cover the cost of the AL interactions. This budget is instantiated by the strategy or by the designers of the learning system. The majority of the literature in AL assumes the budget to be the maximum number of queries that can be posed to oracles; often, this is infinite. An alternative is, for example, the work of Vijayanarasimhan et al., where the budget is a limited amount of time to spend on annotation [VJG10]. The AL budget can be interpreted as the stopping criterion for an AL strategy, i.e. the method by which the system stops asking questions. However, different stopping criterion, unrelated to the budget, might be used as well or instead. For example, in [Vla08, BVS09, OT09], the system stops querying oracles when the latest new labelled data do not improve the performance of the learner.

The oracles

AL strategies also need to be aware of the nature of the oracles. In ML scenarios, most of the active learning literature assumes the oracle to be omniscient so that she is always able to provide the correct answer for the questions being asked. But this paradigm usually fails to hold in real-world applications, most of the time because oracles are human experts in the domain of the problem at hand. Humans are fallible; they can make mistakes during the labelling process, despite their expertise in the task, introducing errors or noise in the system (e.g. labelling handwritten digits might be a difficult task for human annotators). In the extreme case, they could even be dishonest or biased in their answers. There might also be situations in which an oracle gives an answer with a certain amount of uncertainty; or she may also not be able to provide an answer (or a complete answer) to the query, due to a lack of expertise, for example.

The ML literature also generally assumes that questions are asked to a single (and always the same) oracle. But a possible and more general scenario is instead the one where more oracles are assigned to the labelling process. For example, hiring multiple oracles might be less costly than having a single oracle; or it might be a better choice for the particular task at hand; or a single oracle might be unable to answer all she is asked due to a lack of expertise. Employing multiple oracles introduces new opportunities and challenges for AL. Opportunities because it is now possible to assign different data points to be labelled by different oracles, based on their level of expertise or their labelling cost, for instance. Also, as said before, it could be cheaper to employ multiple oracles

instead of a single one; an example of this is the crowdsourcing mechanism used in Amazon Mechanical Turk. Leveraging multiple oracles can also help label 'difficult' instances for some domains (e.g. in the radiology field, specialists often disagree on the diagnosis for the same radiological image). In this latter case, the system can query more than one oracle with the same data point and combine their answers afterwards. At the same time, having multiple oracles in the AL framework poses new challenges: problems arise such as the possible inconsistency across oracles, e.g. some of them might be more reliable than others in general; some of them might be less accurate or biased than others depending on the query being asked; there is also concern on how to combine answers from different oracles in cases where more than one is assigned the same data instance to label and when they disagree on the answers.

Finally, in an ML scenario, oracles are considered to be consistent: if asked twice the same question, they are expected to answer in the same way. Some exceptions might be when an oracle increases her expertise, so that her response accuracy increases as well; or when an oracle makes a mistake, either in the first or second time that she is asked the question.

AL strategies that relax these oracles-related assumptions have been proposed. For example, Donmez & Carbonell presented a decision-theoretic approach where imperfect multi-oracles are queried. In their work, oracles might refuse to answer, make mistakes and charge different fees to label queries [DC08]. In [DL10], Du & Ling study the impact of a noisy oracle. Other examples are [YRFD11, ZC15] where the AL strategies model the oracle's expertise in a multi-oracle framework [YRFD11].

Different ways of querying

Alternative ways of querying an oracle have also been explored in ML scenarios. For example, feature-based Active Learning, where the oracle is asked to supply the values of features, has been applied in settings where the labels of the instances are known but many feature values associated with the instances are unknown (e.g. in medical applications). In what is sometimes called *active dual supervision*, the AL can ask queries to elicit either feature values or labels, e.g. [Set11, DSM09]. Another example is given by Raghavan et al.: they proposed a text classifier which interleaves queries for labels with feature-based queries [RMJ06].

In other settings, especially in tasks where instances are synthesized for train-

ing, rather than being naturally-occurring examples, AL strategies might pose questions in terms of *class labels*. In this case, the oracle is asked to provide examples of instances of a particular class, in a sort of reverse of classical Active Learning, e.g. [LBA+07].

In domains such as protein family modelling, image retrieval, stock market prediction, and drug activity prediction, instances might be naturally organised into "buckets". Each of the buckets will be assigned a label and each instance inside a particular bucket has the same label. In this scenario, Multi-Instance (MI) Active Learning has been applied: oracles are asked to assign a label to a set of instances rather than to a single instance, e.g. [SCR07].

4.3 Active Learning in Recommender Systems Scenarios

In an RS scenario, personalized recommender systems acquire user profiles (e.g. in terms of ratings), and from those profiles, they build a model. Using the model, they provide recommendations to individual users. Other things being equal, the better the profiles, the better the model; and the better the model, the better the recommendations. Growing a user's profile is therefore of vital importance for a recommender, especially in situations where there is an evident lack of data, i.e. to solve the specific problem of *cold-start users* — new and low-activity users whose profiles contain little data.

Profiles are typically populated initially during a sign-up process, where a new user states her preferences for a small subset of the items. Some systems may choose this subset of items at random and ask a user, of the ones she is familiar with, to rate as many as she wishes, maybe until a certain minimum profile size is reached. Other systems allow users to freely search or browse the item catalogue instead. Subsequently, profiles grow either by observing the user's interactions with items (in the case of implicit ratings) or by acquiring the user's opinion of an item after she has consumed it (in the case of explicit ratings).

One way to intelligently acquire new informative data, i.e. data that would enrich the quality of the profiles, is to employ Active Learning, where the recommender system proactively solicits the user's opinions for selected items. AL strategies might be deployed during sign-up or during a user's regular use of the recommender. Similarly to ML scenarios, AL in RS scenarios must make a

utility-cost trade-off by maximizing the quality of the recommendations while minimizing the cost of obtaining new data. So far, in the RS literature, AL has been employed to solve the cold-start problem (e.g. [ME19, ZYZ11]). However, AL's generality suggests that it could also be employed for more mature users (by which we mean users who have more data in their profiles), e.g. to regularly acquire fresh data and stimulate new recommendations. (Indeed, in Chapters 5 and 6 of this dissertation we study AL beyond the cold start scenario.)

While AL's goals are very similar for ML and RS scenarios, the characteristics of their frameworks are different: in the following, we give details about such differences.

The oracles

In an RS scenario, the assumption often made in ML of a single omniscient oracle is lifted. In real-world RS applications, oracles are also humans, i.e. typically users who joined the recommendation platform. But, importantly, they are not considered to be experts in the domain of the recommender, at least not in the same way as they are considered experts in an ML scenario. For example, the average user of a music recommender is not necessarily a musicologist or a music expert but, usually, she is just a music consumer (listener). The most one might assume is that the user is an expert in the matter of her own tastes, although even this is open-to-discussion.

In AL for RS, users are typically asked to provide ratings for items from the recommender's catalogue. Their answers strongly depend on their preferences, their knowledge, or their previous experiences with such items. Informally speaking, it turns out that oracles in RS provide *subjective judgments* on the queried items, i.e. judgments influenced by personal tastes, rather than the 'objective' judgments provided by the expert oracles in ML. For example, if a classifier asks the oracle to classify whether a certain image contains a cat, while there may be uncertainty, there is only one correct and 'objective' answer for the label (i.e. "yes, the image contains a cat" or "no, the image does not contain a cat"). On the other hand, if a movie recommender asks a user for a rating for the movie "Titanic", there is no 'correct' rating: the user gives her personal subjective opinion on the movie. Despite this distinction between ML and RS, the RS literature does include work in which attempts are made to identify and correct user ratings that are considered noisy (e.g. incorrect, biased, or unreliable) — although this work has not influenced the AL literature. For example,

Pham et al. propose a methodology to identify incorrect ratings (i.e. ratings that do not reflect a user's preferences) and subsequently correct them by leveraging a group of 'expert' users (nominated based on the users' preferences) [PJN12]. Adomavicius et al. study a way of correcting user ratings affected by bias introduced by the system's recommendations; they propose a user-interface which supposedly reduces such bias for a user when rating an item [ABSZ14a].

Related to the subjective nature of user ratings is the issue that user consistency is not always ensured. Indeed, users' preferences and opinions may vary over time. For example, a user may give a 5-star rating to the movie "Titanic" at first. After six months, if asked again, the same user may give a 4-stars rating because she has experienced many better movies in the meantime, which re-calibrate her judgement of "Titanic". In studies such as [HSRF95] and [CLA+03], for example, it has been shown that users are quite consistent in their ratings when re-rating after six weeks [HSRF95] but less consistent when re-rating occurs later [CLA+03]. It may not be just subsequent experiences that influence a user's judgements over time: recency plays a part, for example. Consistency, in general, may depend on factors such as the use of different rating scales ([CLA+03]) or the number and identity of items that a user is asked to rate together.

In RS scenarios, users are not obliged to provide an answer when queried by an AL system: they may refuse to answer (e.g. for privacy reasons); or they may not know the answer to the question. A common situation, indeed, is when a user is asked to provide a rating for an item that she has not consumed or purchased yet. For example, if asked to rate the movie "Titanic", a user may not be able to do it because she has not watched it yet. Therefore, in selecting items to ask about, a good AL strategy must choose items, not just on the basis that ratings for these items will improve the quality of the recommendations, but also on the basis that they are ones that it predicts are familiar to the user, since there is little point in asking about items the user is unlikely to be able to rate. Therefore, some strategies try to maximize the probability that the items being queried are familiar to the user. A strategy based on item popularity, i.e. selecting the most popular items (the ones which have the highest number of ratings) is probably the simplest strategy to achieve that. However, even if this strategy usually collects lots of ratings, it has been shown that these are mainly positive ratings because popular items are most of the time liked by the majority of the users. It turns out that acquiring positive ratings for popular items usually brings little additional discriminative information to the system

[ERR16], i.e. it leads to little improvement in the final recommendations. An effective alternative strategy is proposed by Elahi et al., named Binary Predicted [ERR14]. It leverages a factorization model (built on binarized data) to predict which items have been consumed by the users (and therefore which items can be rated by users). Another example can be found in [EBRT13], where Elahi et al. aim at the same goal by leveraging a factorization model enhanced by side data about the users (e.g. gender, age group and the scores for the Big Five personality traits [MJ92]). Finally, in [RAC+02], Rashid et al. ask a user to rate the items with the highest similarity to the user's previously-rated items, with the assumption that the user is familiar with items like those that she has rated before.

Obvious too from the previous discussion is that in AL for RSs, there are multiple oracles rather than a single oracle. In truth, to some extent, we could still look at AL interactions in recommenders as many one-to-one interactions between the recommender and each of its users. However, in this dissertation, we choose to consider AL in RS as a multi-oracle framework, because the system also generally has the freedom to query (or not to query) each of its users. The system may even decide to query some users more often than others. Since these decisions are part of the AL strategy, the multiple oracle framework is more suitable to describe active learning in the RS scenario.

The gain, the cost and the budget of AL

As mentioned before, improving recommendation quality while minimizing the cost of obtaining improvement is the main goal of an AL strategy in an RS (and ML) scenario. When querying a user, a good strategy should ask for information that, if acquired, would improve recommendations for the user. However, such information is potentially useful to improve the recommendations of other users of the system: this happens, for example, in collaborative filtering recommenders that leverage all users' data to recommend to a single user. (In fact, in Chapters 5 and 6 we will investigate AL under this perspective as well.) Newly acquired data is also potentially useful to solve the *cold-start item problem* (as opposed to the cold-start user problem), i.e. when a new item is introduced into the catalogue, it will have no or little interaction data. Work in the AL literature that explicitly address this issue includes, for example, [ZLH+20, AAAE+15, ABBC17].

For an AL strategy in an RS scenario, in addition to the cost faced by the strat-

egy to generate the query, there is the cost to the users who have to provide the corresponding answers. To the best of our knowledge, the RS literature always assumes that this cost is proportional to the number of items the user is asked about; we are unaware of any work on modelling these costs in a more sophisticated way. Indeed, in the typical AL scenario, the budget is chosen in a simple, heuristic and single-user way: for example, the system is allowed to ask a given user to rate up to a maximum number of items (i.e. a batch query), where the maximum is chosen heuristically based on thoughts about screen size and the burden placed on users. There may also be a maximum number of users that the AL system is allowed to interact with. In evaluating AL strategies, this maximum number of items and the maximum number of users may be allowed to vary, to assess their impact on RS performance. A more in-depth and interesting investigation on user-related costs is from Cremonesi et al.: they empirically demonstrated (by means of a user study) that the cost for the users (i.e. in their case, the user-perceived effort) is somehow proportional to the number of items they are asked to rate; but, interestingly, global user satisfaction is not negatively affected by the increased effort when balanced by significantly better recommendations [CGT12]. Different cost-budget models have also been considered when the query is posed to the user differently: e.g. pairwise queries, group queries, etc; see below for explanations of these. However, even in this work, costs and budgets are always modelled in terms of the number of queries posed to users; alternative models are not explored.

Different ways of querying

Research has also been done in RS scenarios on AL strategies that pose different kinds of queries to users (i.e. different from asking them to rate a single item). Some work has been done on pairwise preference elicitation, where users are presented with two items and asked to choose which one they prefer, e.g. [BR15, KRG18, SKRdR18]. Other AL strategies provide users with a decision-tree based questionnaire (i.e. a sequence of items that the user can either like, dislike or skip [GKL11], or rate [KNST14]). Others ask users to compare groups of items (and choose the best one), in an attempt to also reduce the user effort in the elicitation process [LHZ14, CHT15]. Finally, there is work on contextual elicitation [BR17], where users are asked to provide, in addition to the rating of the item, some contextual situations (e.g. weather, companion) that explain the given rating.

Table 4.1: Categorization of AL strategies.

Category	Objective
Uncertainty Sampling	Strategies that query for data which the model is most uncertain about.
Expected Error Reduction	Strategies that query for data that will supposedly reduce the future error of the model.
Expected Model Change	Strategies that query for data that will supposedly change the model by the greatest extent.
Representativeness-based	Strategies that query for data which better represent the data space already available for training the model.
Hybrid	Strategies that combine different criteria to build the query.

4.4 A Categorization of AL strategies

Different AL strategies take different approaches to selecting what to query. In this section, we review examples of strategies that have been proposed in ML and RS scenarios and belong to one of five main categories, summarised in Table 4.1. However, framing all the strategies proposed in the ML and RS literature into one of those categories is a quite hard task because, as we analysed in the previous section, ML and RS are quite different scenarios after all. Our categorisation that we present here focuses on the objectives that an AL strategy seeks, rather than the particular scenario of its application (i.e. ML or RS). In other words, in this section, we look at which kind of data is of interest for a particular strategy, where the approach for querying such data is common in both ML and RS scenarios. Alternative and complementary categorizations are given by Settles [Set12] and by Elahi et al. [ERR16] in ML and RS scenarios, respectively. For the sake of the exposition, all the strategies we present when exemplifying the categories build the queries in a pool-based sampling fashion. In examples of AL for RS, unless stated otherwise, we consider the common scenario where an AL strategy asks a user to rate items drawn from the pool of unrated items for that user.

4.4.1 Uncertainty sampling

This class of strategies asks questions to get to know information about which the learner is most uncertain. In ML, such strategies select the unlabelled examples which the learner is least certain how to label [LC94]. Uncertainty sampling fits well for probabilistic classifiers for example, because they directly

output probabilities which represent the classifier's confidence about each possible label for a given unlabelled instance. However, this AL class generalises well to other kinds of classifiers; the key point is to have a metric to measure the uncertainty of unlabelled instances. Different works propose different metrics: a least-confident-based metric, e.g. [LG94, CM05b, CPLS20]; a margin-based metric, e.g. [SDW01]; and entropy [Sha48], e.g. [Hwa04].

Uncertainty sampling strategies have been applied to RS too. Similarly to ML, attempts have been made to provide a measure of uncertainty or confidence for the predictions of a recommender: useful overviews about this topic can be found, for example, in [BGOZ18, ZOBG18]. Concretely, confidence metrics are based on a few simple non-personalised approaches, sometimes being combined (e.g. [HBOT13]). An item's popularity is one of those: when giving a confidence score for a particular user-item pair prediction, practitioners use the number of ratings associated to the item (i.e. the more popular is the item, the higher is the confidence on the prediction) [MLG+03, AKK07]. Thus, selecting popular items for the query aims at acquiring many ratings (since many users are likely to known about popular items) [RAC+02, TCRC02]; selecting unpopular items aims at reducing the uncertainty of the predictions on those unpopular items.

The item rating variance confidence score is another metric used to measure uncertainty. It relies on the intuition that items rated diversely by the users are controversial items; therefore, making accurate predictions on those items is difficult and recommending those items is risky. Works such as [AKK07, Maz13] leverage rating variance to refine recommendations of collaborative filtering algorithms; [KM01, TCRC02] use it as an AL strategy: it selects the items with the highest variance for the queries. Entropy scores are also used in RS by AL strategies, where items with the highest entropy are selected to be rated by users [KM01, RAC+02, BZM03, RKR08].

Besides being non-personalised, item popularity and item rating variance are independent of the recommendation algorithm used to predict and recommend. This makes such metrics unlikely to provide reliable scores for different kind of recommendation models. Differently from ML applications and to the best of our knowledge, a recommender's model whose outputs explicitly and intrinsically include personalised confidence scores for the recommendations and predictions provided to its users has not yet been proposed. However, Koren and Sill's work [KS11] is one attempt to provide personalised confidence scores

(calculated afterwards) from the model's predictions. Indeed, their OrdRec recommender outputs a full probability distribution of the expected ratings for each user-item pair, rather than only a single rating prediction. Afterwards, confidence level scores are associated with the amount of concentration of those rating distributions, i.e. in terms of standard deviation, entropy, or Gini impurity. Confidence scores so defined are therefore personalised and algorithm-dependent.

Uncertainty sampling strategies are widely-used because they are usually easy to implement and computationally cheap when building the query. For these reasons, they are particularly well-suited to the AL pool-based sampling framework where typically lots of candidates need to be evaluated in order to compose the query. However, they are a somewhat heuristic approach because they do not guarantee performance improvements for the system. For example, in a collaborative-filtering recommender, querying items with high rating variance might be querying controversial, unpopular items that do not guarantee to improve future recommendations (i.e. because ratings for such items weakly affect the predictions of the ratings of other items).

4.4.2 Expected Error Reduction (EER)

This class of strategies tackles the problem of selecting the most informative data points for the learner by directly looking at the expected performance of the model. EER strategies build queries that, once the answers are obtained, will most likely reduce the error of the model on its task. One problem is that the learner does not know what the answer of the oracle will be beforehand. Additionally, even if the strategy knew in advance the oracle's answer, the future error of the updated model will be still unknown; that is why EER strategies seek to minimize the *expected value* of such future error, rather than the *real unknown value*.

Concretely, when evaluating one candidate data point for the query, the general idea is to estimate the expected future performance of the model were it to be trained on the training set augmented by the instance that comprises the candidate and its elicited label. Because the true label of the candidate is not known, the calculation is usually done using expectation over all possible labels under the current model. Alternatively, but often more expensively, all different instance-label combinations are evaluated. This procedure is iterated over all the candidate queries, and the one showing the best-expected performance is

selected. Examples that use the EER approach in the ML scenario are the Naive Bayes text classifier in [RM01]; the SVM classifier used for host intrusion detection in [MNS+07b]; and the Gaussian random field model for handwritten digit recognition and text classification tasks in [ZLG03].

In the RS scenario, EER strategies are designed on the same principles. Indeed, queries are built in order to improve the expected predictive accuracy of the recommender at hand (or to reduce the error of its predictions). An example is by Golbandi et al. [GKL10], where the authors design an AL strategy for interviewing new users to improve a factorized item-item model [Kor10]. In subsequent work, the same authors explore the use of a decision tree-based AL strategy to interview users [GKL11]. Both strategies select items which minimize the RMSE over the training set. Another example is [SKRdR18], where an optimal pairwise preference elicitation strategy is designed to mitigate the cold-start problem.

EER strategies in both ML and RS share the same pros and cons. Strategies of this kind have the great advantage of being near-optimal in the sense that they do not rely on heuristics; they optimize directly for the utility of the system instead. The utility can be expressed by different metrics (e.g. 0/1-loss, log-loss & ROC curve in ML classifiers, and recall, precision & RMSE in recommenders) with no other modification to the approach at all. The approach is also mostly model-independent: we just need to define an appropriate objective function to be optimized as a selection criterion. The main drawback of EER strategies is their high computational cost: optimizing a utility function is generally costly, and this has to be often done separately for each of the data instances in the candidate pool. Where possible, incremental training can be used to mitigate this problem (at the probable expense of the model's accuracy).

4.4.3 Expected Model Change (EMC)

The intuition behind this class of strategies is that the instances (once labelled) that are able to produce the greatest change in the current learned model are the most informative ones — or, at least, are potentially the most informative. EMC covers many strategies, each tied to the different model or models being used for a particular task. In ML, for example, the EMC approach is applied to improve performances of models where gradient-based training is used [SCR07, SC08]. In such settings, the instance that would produce the greatest change in the gradient of the model's objective function is selected for the query.

In RS, an example is the method designed by Rubens and Sugiyama [RS07], in aid of a user-based collaborative filtering model. They propose an influence criterion that measures the effect that rating a candidate item potentially has on the predicted ratings of other unrated items, i.e. because each new rating has a potential impact on the user-neighbourhood calculations of the model. The candidate item with the largest expected change in the predictions of the unrated items (and therefore in the model's user-neighbourhoods) is selected for the query.

Similarly, Mello et al. seek to query for ratings that most impact the model's predictions for other unrated items, where such predictions are made by a user-based bipartite graph model [MAZ10]. In their strategy, the item whose rating will impact the highest number of predictions is selected for the query.

Similarly to uncertainty sampling, EMC's heuristics do not establish a link between queries and performance improvements, i.e. changing the model does not ensure a boost in performances. Also, like for EER, because the strategy does not know in advance the label that will be attached to the query, the strategy has either to evaluate the impact of all possible labels on the model (which is computationally expensive) or to ignore the labels that might be obtained by the data point queried (like in [RS07, MAZ10] for example).

4.4.4 Representativeness-based sampling

Uncertainty Sampling strategies suffer from the fact that outliers or unrepresentative data instances are often selected for the query. This is because the approach reasons about the uncertainty of single instances only. EER and EMC are different because they look at the impact of both the candidate for the query and the rest of the labelled data available on the *aggregate*. Thus, unrepresentative or outlier-like queries are avoided. This idea of *representativeness* is the key ingredient of a set of strategies whose goal is to query for data that well resemble the characteristics of the data already available for training. In ML, examples of this kind are density-weighted strategies. Their general goal is to select one or more among the most informative queries (by using one of the previously presented strategy classes) but at the same time weighting the choices by favouring those which resemble the characteristics held by the training data distribution [SC08][1]. Other examples can be found in [FTIT98, MN98, NS04].

[1] To note, this kind of strategy might be considered as a hybrid approach, but in this dissertation, we decide to describe it in this section to enforce the importance of the representativeness-

In RS, one strategy of this kind is co-coverage [GKL10], that selects highly co-rated items, which are likely to be ones that can improve the accuracy of a factorized item-item model. Another example, by Liu et al. [LMLY11], uses a modified matrix factorization algorithm, which they call Representative-Based Matrix Factorization (RBMF), to first identify the most representative users and items in the system. Then, as an AL strategy, representative users and items are used as targets from which to elicit ratings to solve the new item and new user problems, respectively.

On the one hand, employing representativeness-based strategies can result in queries that avoid selecting outliers (in ML) and obscure or rare items (in RS), both of which might lead to acquiring little informative data. But, on the other hand, it might turn out that such strategies bring little or no explorative value in their queries. This is because representativeness-based strategies exploit the characteristics and the distribution of the data already available, giving less scope for obtaining new diverse and heterogeneous data.

4.4.5 Hybrid

A single criterion, i.e. like each of the ones presented so far, may fail to successfully identify the data instances that are worthy of being labelled. For example, attempting to reduce the uncertainty of a model is simple and cheap but might lead to selecting outliers or unrepresentative data; selecting data which directly minimizes the future error of the model is a nearly-optimal approach to AL, but it comes with expensive computation costs; EMC strategies might be expensive and are heuristic; representativeness-based strategies are also heuristic and may result in lack of exploration in the data space. For these reasons, it is natural to think about designing new strategies which are combinations of two or more strategies in an attempt to obtain the best features brought by each of the individual methods.

Hybridization can be obtained in different ways. Voting-based hybridization is one solution, i.e. each individual strategy votes for data instances in the pool and the instances with most votes compose the queries, e.g. [ERR11]. Another solution is where each individual strategy scores the data instances in the pool, and those scores are combined (e.g. by a linear combination) to produce a final score which is used to build the query. In ML, some hybridization examples are [DCB07, HRJL08, HJZ10, XYT+03]. Examples in RS are

based criteria when building a query.

[ERR14, ERR11, RAC+02, RS07, RKR08, KFNS11, ZYZ11]. Although hybrid strategies can be very effective in selecting useful data for the learner, the associated computational cost might be problematic; parallelisation, when possible, might be used to mitigate this problem. Also, deciding how many and which strategies will be part of the hybrid and how to practically combine and tune their output might be difficult.

In this chapter, we have described the principles of Active Learning, framing their application in RS scenarios in addition to the classical ML scenarios. To the best of our knowledge, for the first time in the literature, we also proposed a review that explores differences and similarities of AL goals and approaches through a comparison between ML and RS scenarios. In the next chapter, we review methodologies for evaluating AL strategies and we propose our own alternative.

Chapter 5

A New Active Learning Evaluation Framework

As with the evaluation of recommender systems, preliminary evaluation of AL strategies for recommender systems must be done offline, with some simulation on a pre-collected dataset. Promising strategies can be further evaluated both in user trials and online in, for example, A/B experiments within a deployed recommender system. The offline evaluation can help to narrow the number of strategies that need to be evaluated in costly user trials and online experiments. Essentially, the offline evaluation involves simulated users who, when prompted by the AL strategy, may reveal ratings that were previously hidden from the recommender system.

Where the literature describes an offline evaluation of Active Learning, the evaluation is relatively narrow and incomplete: mostly, the focus is cold-start users (see Section 4.3); the impact of newly-acquired ratings on recommendation quality is usually measured only for those users who supplied those ratings; and impact is measured in terms of recommendation accuracy only. Moreover, the traditional AL offline evaluation methodology does not take into account the bias problem in RS offline evaluation. As shown in Chapter 3, the use of a biased dataset affects the results of an RS evaluation. However, we argue, it also affects the evaluation of AL strategies as well.

In this chapter, we describe the new offline evaluation methodology that we are using to evaluate Active Learning. One of its core features is that it mitigates the bias, which facilitates a more authentic picture of the performances of the AL strategies under evaluation. We do so by means of our intervention approach,

presented in Chapter 3.

Additionally, we argue that:

- Active Learning may benefit *mature users* (by which we mean users who have more data in their profiles), as well as cold-start users;
- in recommender systems that use collaborative filtering, the newly-ac quired ratings may have an impact on recommendation quality even for users who did not supply any ratings; and
- the new ratings may have an impact on aspects of recommendation quality other than accuracy (such as diversity and serendipity).

Differently from the narrow classic offline AL evaluation, our more comprehensive offline method can assess the impact of an AL strategy concerning these three aspects.

In Section 5.1, we describe the solutions provided by the literature to evaluate Active Learning, also introducing the terminology that will be used later on. Building upon this review, in Section 5.2, we present our new comprehensive framework to evaluate Active Learning. In Section 5.3, we present a case study to compare our new evaluation methodology to the classic one, to show the greater analysis potential brought by the former. We draw conclusions about the chapter's content in Section 5.4.

5.1 Classic Offline Evaluation

5.1.1 Background

In early work, AL was used to improve recommendations either for users who had small user profiles (i.e. cold-start users) or new users (sometimes known as extreme cold-start users). For these new users, AL has been used in the sign-up process to assist them in building their initial profiles (see Section 4.3). However, AL's generality suggests that it could also be employed for more mature users (by which we mean users who have more data in their profiles), sometimes referred to as 'warm users', e.g. [XYT+15]. For these users, AL might be used, for example, to regularly acquire fresh data and stimulate new recommendations. To the best of our knowledge, only Carenini et al. [CSP03] and Elahi et al. [ERR14] consider non-cold-start users in the AL scenario. Carenini et al. describe a conversational approach to be used *after* the sign-up process. How-

ever, even this work is quite narrow since its evaluation is confined to users who have 50 ratings; no other scenarios are considered. By contrast, in the work of Elahi et al., the AL strategy is applied to all the users, irrespective of the sizes of their profiles. Therefore, Elahi et al.'s evaluation includes cold-start users but more mature users too. Unfortunately, they do not then break their results down. Therefore, we cannot discern whether a strategy has a different impact on cold-start users from the one it has on more mature users. Instead, our methodology, described in Section 5.2, addresses this, allowing one to measure Active Learning's benefits on both cold-start and mature users.

Elahi et al. also make the point that all of the offline evaluations of AL strategies that are reported in the literature (apart from their own) take what they call a *user-centric* approach: results are reported only for the subset of simulated users who participate in the AL interactions (typically again, just the cold-start users). Elahi et al. pioneer the idea of a *system-wide* evaluation, in which they report the impact of the AL on the whole user population. This is important in recommender systems that use collaborative-filtering where new ratings might influence the recommendations made to other users, not just the users who participate in the AL. In practice, both user-centric and system-wide perspectives must be measured: there is a time cost, and perhaps a cognitive cost, to provide a rating, so those who participate must generally benefit, but those who are not participating should also benefit or, at least, see no worsening of performance. Thus, in our proposed evaluation, we break down the sets of users even further.

We also note that, in early work, the impact of the AL strategy on the quality of the recommendations is measured by computing prediction error on the test set using metrics such as MAE or RMSE, e.g. [RKR08]. In the same way that prediction error has been displaced in recommender system evaluation by metrics such as Precision, Recall and nDCG (see Section 2.1), this has happened in the evaluation of AL strategies too, e.g. [ERR14]. A few of the evaluations that are reported in the literature employ other metrics such as Average Popularity [FTBE$^+$16], Coverage [ERR14], and Spread [FTBE$^+$16, PQEC17]. But it is also recognized that satisfaction nowadays with recommendations is not just a question of their accuracy [MRK06]. It may be desirable for a set of recommendations to be diverse or for the recommendations to be serendipitous. A wide range of metrics has been proposed to measure these 'beyond-accuracy' aspects of recommendation quality, especially for offline evaluation (see Section 2.1). To the best of our knowledge, none of the evaluations of AL that are reported in the literature has ever used these measures. For this reason, in

our proposed methodology, we aim at measuring the impact of AL in terms of 'beyond-accuracy' qualities, as well as accuracy.

Finally, to the best of our knowledge, there is no work in the literature on debiasing an offline Active Learning evaluation. However, we argue that if one wants to obtain reliable results from the offline evaluation of an AL strategy, the bias in the data should be mitigated. Thus, our proposed AL framework that we describe in Section 5.2 makes use of WTD_H intervention to provide a debiased evaluation.

5.1.2 Setup and notation

Offline evaluation of recommender systems typically involves a snapshot, measuring how one or more models (built from training data) perform on one or more test sets. Offline evaluation of AL strategies for recommender systems, on the other hand, requires that we simulate the behaviour of users and measure recommender performance both before and after the application of the strategy to see how performance changes. This requires the dataset to be split into at least three parts: the known ratings, the hidden ratings and the test ratings. The *known ratings* are the ones on which the initial recommender model is built; the known ratings are also the ones on which the AL strategy generally operates. The *hidden ratings* are the ones which the simulated user might reveal to the system if prompted to do so by the AL strategy. Subsequently, these elicited ratings can be added to the known ratings, and a new recommender model can be built. The performances of the initial recommender model and the new recommender model are measured against the ratings in the test set, i.e. the *test ratings*. We refer to this as a single-iteration AL evaluation (see Section 4.1), since it measures the effect of a single application of the AL strategy, and this is the focus in this dissertation. In Chapter 7, we discuss multi-iteration AL evaluation, which simulates more than one application of the strategy.

We will now formally describe the typical offline evaluation framework, employed by most of the works in the literature. The setup that we describe captures the most typical scenario for the evaluation. However, some small differences might apply to this protocol but leaving unchanged its core features.

How to prepare the dataset for the evaluation

The first step consists in preparing the dataset for the experiment's execution. As in most of the literature, we will assume a dataset of explicit ratings (see our statement in Section 2.1). However, the principles of this setup easily adapt to implicit ratings datasets. We recall the notation introduced in Section 2.1, where we indicate with D such a dataset of explicit ratings. We denote with $r_{u,i} \in D$ the rating of the user-item pair (u, i): $r_{u,i} \in \{1, 2, 3, 4, 5\}$ (i.e. ratings are in a 1 to 5 stars scale) if the rating is observed, $r_{u,i} \notin D$ if $r_{u,i} = \perp$ otherwise.

We start by eliminating users who have too few ratings (i.e. each user must have enough ratings so that we can meaningfully partition them for the purposes of the experiment) or too many ratings (i.e. to avoid outliers).

To assess the benefit of Active Learning, we want to evaluate a single AL iteration, i.e. performance before and after using the AL strategy. If we do this for just one user, we are unlikely to see any impact. In even the best of circumstances, so few new ratings will be acquired that the difference in recommender performance will be negligible. So, we randomly select a *group* of users to whom the AL strategy will be applied. Let's call these the Active users, $U_{Active} \subseteq U$. The choice of how to sample U_{Active} depends on the objective of the AL investigation. For example, in [ERR14] we have $U_{Active} = U$, so that every user is queried by an AL strategy: this is done to consider the evolution of the global system performance under the application of an AL strategy applied to all the users. In [HY08] and [CSP03], by contrast, the set of Active users is a subset of the total users in the system: in the former, Harpale et al. simulate a cold-start scenario for Active users; in the latter, Carenini et al. simulate a scenario where Active users have already been through the initial sign up process (i.e. mature users).

Next, we randomly partition the ratings in D of the Active users into three parts: K (the known ratings), H (the hidden ratings) and T (the test ratings). On this point, the literature is divided into those who generate the three parts by means of a random user-based split, e.g. [HY08], and those who use a simple random split e.g. [ERR14]. The user-based split ensures each user has a fixed proportion of her ratings (and the same for all users) in each of the three partitions, e.g. 60% in K, 20% in H, 20% in T. The simple random split will randomly partition the ratings, e.g. 60% in K, 20% in H, 20% in T, with no user-based constraints. In this dissertation we choose to consider the simple random split alternative (and we will make use of it in Section 5.3): the reasons will be explained

in Section 5.2. To complete the setup, we define $K^{before} = \left(\bigcup_{u \in U_{Active}} K_u\right) \cup \left(\bigcup_{u \notin U_{Active}} D_u\right)$ with K_u and D_u being u's ratings in K and D respectively. K^{before} is the entire set of ratings known by the recommender at the initial stage of the experiment and includes all the known ratings of the Active users and all the ratings of the non-Active users available in D.

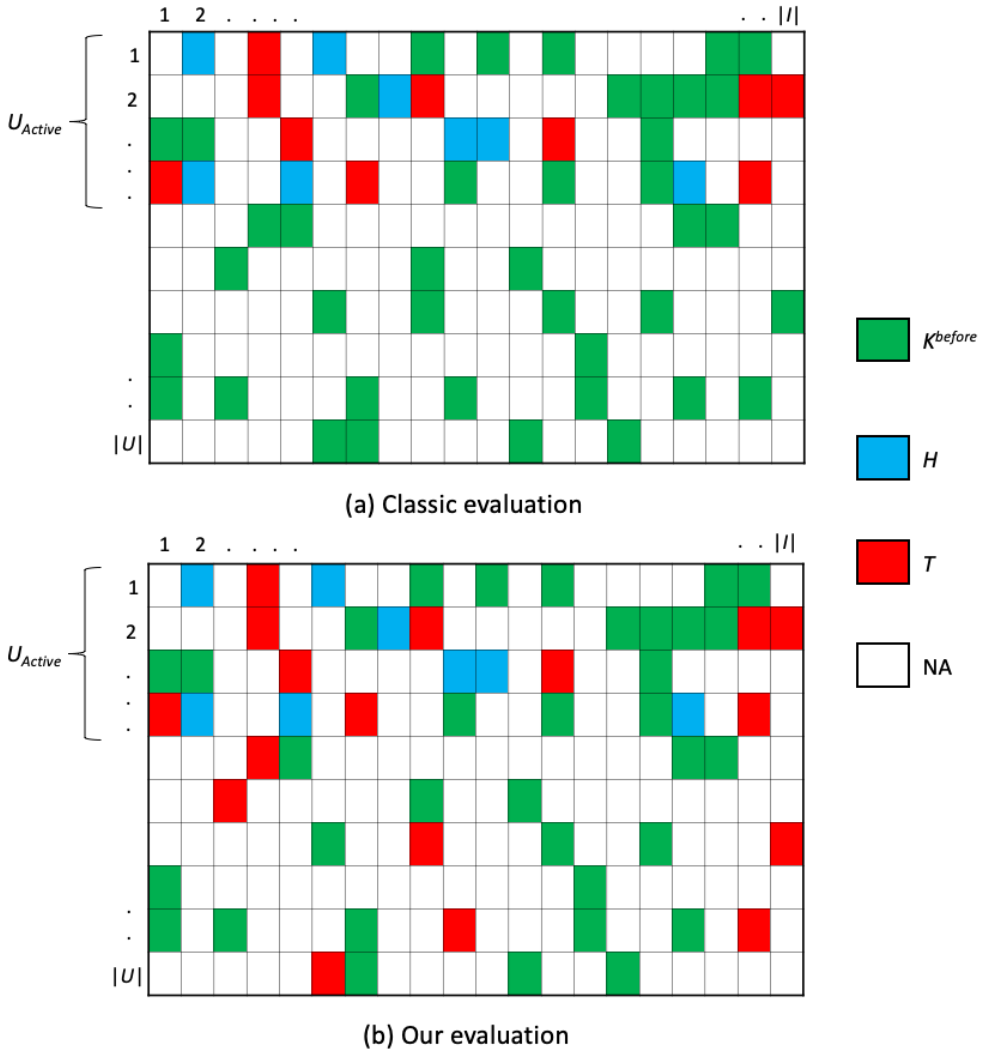

Figure 5.1: High-level visualization of the dataset split for the classic and our new evaluation methodologies. NA stands for Not Available for the experiments, i.e. a rating that is either not available in D or discarded when debiasing the data (see Section 5.2 for the latter reason). In the classic evaluation (a), Active users have ratings in K^{before}, H and T; non-Active users have ratings only in K^{before}. They have no ratings in H (they are not queried by the AL strategy), and none in T (the recommender's performance cannot be tested on such users). Instead, in our evaluation (b), non-Active users also have ratings in T, so that the recommender's performance cannot be tested on such users. In both splits (a) and (b), U_{Active} are randomly-selected from U. The figures simplify by representing them as if they occupy the first rows of the matrix.

In Figure 5.1, we illustrate graphically an example of how the dataset matrix D looks after the described partition.

An AL iteration

Algorithm 1 formally describes how an AL strategy composes the query for an Active user u. We define an AL strategy S_{AL} as a function $S_{AL}(u, n_q, C_u, G)$ of four arguments, such that it returns a query Q_u for the user u. The query Q_u is a set of items (of size n_q, i.e. the budget available to S_{AL}) to be eventually rated by the user u (i.e. in a batch query fashion). The items that S_{AL} returns are selected from a pool of candidate items C_u, i.e. this is pool-based AL, where $C_u \supseteq Q_u$.

Two main alternatives have been employed in the literature to identify the *candidate items set* C_u for the user u. We call these the artificial setting and the realistic setting.

- **Artificial setting**: In some works (e.g. [SKRdR18, JS04]), these candidate items are ones that have a corresponding rating available in the hidden set, i.e. $C_u = \{i \in I \mid r_{u,i} \in H_u\}$; this means candidates are items whose ratings can be successfully elicited in the offline evaluation. Although this is an unrealistic setting because it implies that the AL strategy knows in advance the hidden preferences of the user, it can be very much useful in the design of new AL strategies: indeed, we will make use of it in some experiments of Chapter 6 (and its advantages will be clear at that point).
- **Realistic setting**: A more realistic setting is when these candidate items are ones for which u's rating is not currently known by the recommender, i.e. $C_u = \{i \in I \mid r_{u,i} \notin K^{before}\}$. In our analysis and experiments of this chapter we consider this second alternative when referring to C_u.

The strategy S_{AL} also has a parameter G, which designates the data that the AL strategy can use when selecting Q_u from C_u. Most commonly, G is simply the set of ratings that are currently known by the recommender, i.e. $G = K^{before}$. It is conceivable, however, that G contains knowledge from other sources too (see Section 4.1), for example user personality data [FTBE+16]. The only constraint is that, of course, it must be knowledge that the system has acquired before this AL iteration.

At the core of a strategy S_{AL}, there is a function $f_S(u, i, C_u, G)$, which is responsible for giving a score to a candidate item $i \in C_u$. Indeed, in our framework,

Algorithm 1 Query Selection algorithm for an S_{AL}

Input: u, n_q, C_u, G
Output: Q_u, where $|Q_u| = n_q$
1: $Q_u \leftarrow \{\}$
2: **while** $|Q_u| \leq n_q$ **do**
3: $\quad i^* \leftarrow \arg\max_{i \in C_u} f_S(u, i, G, Q_u)$
4: $\quad$ delete i^* from C_u
5: $\quad Q_u \leftarrow Q_u \cup i^*$
6: **return** Q_u

different S_{AL} will have different $f_S(u, i, G, Q_u)$, and therefore different ways of scoring a candidate item. In Section 5.3 and in Chapter 6, we will define an AL strategy S_{AL} by giving the expression for this f_S. Referring to Algorithm 1, the items filling Q_u are iteratively chosen in the loop of lines 2-5. In each loop iteration, f_S assigns scores to all candidates items in C_u and the item with the biggest score is picked to be included into Q_u and removed from C_u. (Of course, in practice, when f_S is independent of Q_u, i.e. $f_S(u, i, G, Q_u) = f_S(u, i, G)$, scores can be assigned only once for all the items in C_u, rather than being assigned afresh on each iteration.) Of course, our framework provides just one of many possible ways of defining a S_{AL}: different frameworks might consider, for example, giving a score to a group of items instead of single items; or picking the items with the minimum score to be added to Q_u; or not using scores at all.

In an online evaluation, the items in Q_u would be presented to the user u, and she would be invited to rate as many of them as she cared to. In offline evaluation, this is where we use the hidden set of the user u, i.e. H_u. The simulated user provides to the system her hidden ratings for items in Q_u. We will call $E_u = \{r_{u,i} \in H_u \mid i \in Q_u\}$ the elicited ratings of the user u. Note that, by this approach, if the simulated user has the rating, she supplies it. This may seem unrealistic: real users are likely to ignore at least some of the requests made by an AL strategy or, even if willing to engage, may not be familiar enough with the items Q_u and thus unable to provide ratings. This is anticipated in the design of this offline evaluation (at least in the so-called realistic setting) because the overlap between the items in Q_u and the ones whose ratings are available in H_u will typically be small, and often there will be no overlap.

Now that we have established the notation, we can show how to evaluate the impact of an AL strategy S_{AL}. We must measure recommender performance before and after the acquisition of ratings by the strategy, as follows:

Before:

1. Build a recommender model from K^{before}.
2. Test the model (see below).

AL interaction:

1. For every Active User $u \in U_{Active}$,
   (a) Build the query Q_u by means of S_{AL}, i.e. $Q_u = S_{AL}(u, n_q, C_u, G)$.
   (b) Obtain the ratings for these items $E_u = \{r_{u,i} \in H_u \mid i \in Q_u\}$
2. Update the recommender with the elicited ratings $K^{after} = K^{before} \cup \bigcup_{u \in U_{Active}} E_u$.

After:

1. Build a new recommender model from K^{after}.
2. Test the new model (see below).

Testing the quality of the recommender (above) is the same both before and after the AL interaction. For every user $u \in U$, we use the recommender model to obtain a set of recommendations for u and then compute the value of each evaluation metric by comparing the recommendations with u's test set T_u.

Note that where there is more than one strategy to compare, the activities labelled *Before* are run just once. But the *AL iteration* and *After* activities are run once per strategy. All this is to ensure every strategy is assessed in the same conditions before applying their elicitation phase.

5.2 A More Comprehensive Offline Evaluation

In this section, we present our new AL offline evaluation, introducing a few changes with respect to the traditional evaluation described in the previous section. We recall that the core of our solution is to have a debiased evaluation that assesses the impact of an AL strategy on different perspectives (i.e. user-centric and system-wide); on different kind of users (i.e. cold start and mature users); and that measures the benefit of an AL strategy not only in terms of accuracy but also in terms of beyond-accuracy objectives.

5.2.1 Partitioning the dataset

After the same preprocessing step on D applied in the classic evaluation, i.e. filtering out users with few ratings or too many ratings (see Section 5.1), we randomly partition the dataset into three parts: K^{before} (the known ratings), H^{he} (the heldout hidden ratings) and T^{he} (the heldout test ratings). Like what we described in Section 5.1, K^{before} is the entire set of ratings known by the recommender at the initial stage of the experiment; but differently from what we described previously, H^{he} and T^{he} are instead heldout sets, used as sampling spaces to generate the final debiased hidden and test sets, H and T respectively (Section 5.2.2). The partition into K^{before}, H^{he} and T^{he} is performed by means of a simple random split. This is more suitable for our methodology than a random user-based split: the latter split would not be suitable for the purpose of debiasing the evaluation itself (see Section 5.2.2).

Another major difference with respect to the classic evaluation is when to select U_{Active}: in our evaluation, we do it after having partitioned D, while in the classic evaluation this is usually done beforehand. Our approach is supported by two reasons. First, we do not want the choice of U_{Active} to affect the partition of D into K^{before}, H and T. Second, partitioning D prior to choosing U_{Active} will allow us to test the impact of AL not only for Active users but also for all the other users, because all users U will have ratings in the test set T.

5.2.2 Debiasing the evaluation

As we showed in Chapter 3, debiasing the evaluation of an RS is a key feature to obtain reliable results for an offline experiment. There, the biased dataset D was partitioned into two MNAR biased parts, i.e. training set and test set, prior to the WTD/WTD_H intervention on the test set. But, AL strategies are evaluated by splitting the dataset into three MNAR parts: the known set, the hidden set and the test set (as we did in the previous section, partitioning the dataset into K^{before}, H^{he} and T^{he}). We could potentially debias any of these three sets.

Debiasing the test set

It is obvious that, if we want an unbiased evaluation, then we must, at the least, debias the test set. To the best of our knowledge based on the literature, AL strategies have never been evaluated offline on an unbiased test set before.

Debiasing the hidden set

In the offline evaluation of AL strategies, the hidden set has a significant impact on the final performance of an AL strategy. We can imagine, for instance, that an AL strategy that asks the users to rate popular items might have success in eliciting many ratings from the simulated users in an offline evaluation if the ratings in the hidden set are skewed towards popular items. However, that does not mean that the same AL strategy would perform as well in practice. Its performance in practice will only be similar to performance in the offline evaluation if opinions that the user has not revealed to the system have the same distribution like the ones in the hidden set.

A user's unrevealed opinions are unlikely to be MAR. Users are influenced by external confounders. For example, a user is more likely to have opinions about items that she has been exposed to, such as items that are popular in general or that have been suggested by her friends. So, her unrevealed opinions are MNAR. But a user's unrevealed preferences are also unlikely to have the same distribution as the ratings in the RS's observed dataset, even though this is also MNAR. This is because, as we have discussed, the observed dataset is influenced by the RS itself. The RS acts as a source of several confounders: the user-interface makes some items more prominent and therefore more likely to be rated; the RS's recommendations are more likely to be rated than items that it does not recommend; and so on.

If we debias the hidden set, we make it more MAR-like, which, by the reasoning of the previous paragraph, is not necessarily correct. However, if we leave it unchanged, then it is distributed like the whole observed dataset D, which, again using reasoning from the previous paragraph, is not necessarily correct. We choose to report results from both, i.e. one set of results where we debias the hidden set (see *INT_HT* below) and one set of results where we do not (see *INT_T* below). True performance should lie somewhere between the two. To the best of our knowledge, our work is the only one in the literature to explore this issue.

Debiasing the known set

Finally, we could debias K^{before} also. We know that, if we build a model on the known ratings without debiasing, then both the model and the AL strategy might be biased; for example, the popularity bias in the data might result in a popularity bias in the recommended items or in the items selected as queries.

However, this dissertation explores the bias in the *evaluation* of AL strategies (and the same was true for the evaluation of an RS in Chapter 3, i.e. there we debiased only the test set) even though we will develop new AL strategies in the following chapters. Our main goal is to show how our approach to unbiased evaluation gives more robust insights into the performance of AL strategies. Therefore, in this dissertation, we will not debias the known ratings set, nor the recommender model or the AL strategies. We will discuss this further in Chapter 7.

WTD_H

In Section 5.3, where we present our experiments (but also in the next chapters), we will use WTD_H (see Chapter 3.3.3) when debiasing hidden and test sets. (We use WTD_H, the version that leverages hypothesized MAR distributions, rather than WTD, because the datasets we employ do not have any MAR portion, see Section 5.3.1.) When debiasing the held out hidden set H^{he} from the previous section, we will generate a final hidden set that we will designate by H; similarly, when debiasing the held out test set T^{he}, we will generate a final test set that we designate by T.

We highlight the fact that our methodology has high generality: a different technique can be used to debias each of these sets: once the held out sets (H^{he} and T^{he}) are generated, any debiasing technique can intervene, without affecting the design of the rest of the evaluation.

In Figure 5.1.b, we illustrate graphically an example of how the dataset matrix D looks after the processes described so far in our proposed methodology.

5.2.3 Measuring the impact of Active Learning

After dataset D is partitioned and properly debiased according to Sections 5.2.1 and 5.2.2, in order to evaluate an AL strategy we will perform an AL iteration in the same way described in Section 5.1. However, differently to a classic evaluation, our methodology allows us to assess the impact of the strategy in a more comprehensive way. Indeed, thanks to the fact that all users in U have some ratings in the test set T, we can evaluate an AL strategy on different perspectives, i.e. we can measure different metrics and report the average values for the following groups of users:

- *All users*, U.

- *Active users*, U_{Active}, i.e. just those users who are randomly selected to be the ones to whom the AL strategy is applied.
- *Respondents*, $\{u \in U_{Active} \mid Q_u \neq \{\} \wedge E_u \neq \{\}\}$, i.e. just those Active users who provide at least one rating.
- *U-less-Respondents*, i.e. everyone who did not provide a rating either because they were not asked for one or because they were asked for ratings but did not provide any.

Furthermore, the way in which we generate K^{before}, H and T will allow us to have both cold-start users and mature users to be tested. Also, every user in U can potentially be in a different scenario, e.g. being a cold-start Active user; or a mature Active user; or a cold-start non-Active user; or a mature non-Active user. In this way, we can assess the impact that AL has on the recommendations for many different kinds of users.

Finally, the quality of recommendations for a single user might be measured with different metrics and values averaged across the different groups of users described so far. In the case study presented in Section 5.3, we will use four metrics, both accuracy and beyond-accuracy.

5.2.4 Setting the values of hyperparameters

Both the recommender model and the AL strategy may have hyperparameters whose values need to be tuned. The AL literature says little on this, and we want to distinguish our work by providing a more rigorous exposition.

We begin by deciding what metric we wish to optimize in hyperparameter tuning. For example, for the recommender model, we might want hyperparameter values that give the highest Recall across all users. For now, we assume that hyperparameters for the AL strategy should also optimize Recall across all users.

Suppose a dataset has been prepared in the way described in 5.2.1 and 5.2.2 so that we have generated K^{before}, H and T. Hyperparameter values must be chosen before running the AL iteration. The only data that can be used for this is K^{before}. If we use any hidden ratings or any test ratings, then we have leakage.

We further split K^{before} in the same way as described in Section 5.2.1 so that we obtain K^{val}, H^{val}, T^{val}, respectively the training set, the hidden set and the test set for the validation phase. To note, we do not perform any debiasing

intervention neither to H^{val} nor to T^{val}. We will justify this when presenting our experiments, in Section 5.3.

We tune the recommender model's hyperparameters first. For each configuration of hyperparameter values, we train a recommender on K^{val}. We measure recommender performance using T^{val}, and we select the hyperparameter values that optimize the metric we chose earlier (e.g. Recall across all users).

Now we can tune the AL strategy's hyperparameters. For each configuration of hyperparameter values, we run an AL iteration by using K^{val}, H^{val}, T^{val}. The AL strategy queries all Active users previously selected, i.e. the ones in U_{Active}. We choose the hyperparameter values that optimize the metric we chose previously (e.g. Recall across all users).

5.3 A Case Study

In the case study that we report here, our goal is not to find the best AL strategy among the five ones evaluated. Instead, our goal to show the potential of our evaluation framework when assessing and comparing different AL strategies.[1]

5.3.1 Datasets

We use the Movielens 1M dataset (ML1M)[2] and the LibraryThing dataset (LT) [CdVR08]. ML1M is one of the most widely used datasets in recommender systems research; it has user ratings on movies. Among the meta-data available for ML1M, we only consider the movie genres, of which there is a total of 18. LT has the ratings that users have given to books, and the tags they have assigned to them. We retrieved a maximum of the 10 most popular tags for every book and kept tags that appeared in the profiles of at least 10 books.

Both datasets have ratings on a 1 to 5 scale in steps of 1 for ML and steps of 0.5 for LT. For test sets, we consider a rating to be relevant if it is above 3 and non-relevant otherwise. We perform the preprocessing step described in Section 5.1 to ML1M and LT, where we remove users with less than 30 ratings and users with more than 500 ratings. In Table 5.1 we report the statistics of both datasets after this preprocessing step. To note, in this case study, we do not

[1] The case study that we report here is an extended version of the one appearing in [CB20a]. Also, the results of the AL strategies that appear here are slightly different from those in [CB20a] due to a small difference in the preprocessing of the datasets.

[2] https://grouplens.org/datasets/movielens/1m/

Table 5.1: Statistics of the datasets

	ML1M	LT
# ratings	∼693k	∼622k
# users	4893	5940
# Active users	1467	1784
# items	3610	37220
avg # ratings per user	∼141	∼104
avg # ratings per item	∼192	∼17
sparsity	0.96	0.998

run experiments with the Webscope R3 and CoatShopping datasets (previously used in Chapter 3). Although they are desirable because they have MNAR and MAR portions, their size is very small in comparison with the ones of ML1M and LT and are therefore not suitable for an AL evaluation.

5.3.2 Active Learning strategies

Of the AL strategies that we have implemented, the ones that appear in this case study are ones that are simple to implement but at the same time useful for our goals. Each of them implements Algorithm 1 for building the queries to the users, and we will give, or describe in words, their different definitions of f_S. We employ:

- *Random* (RND): Active user u's candidate items, i.e. C_u, are assigned a random score in the interval $(0, 1)$ and the n_q items with the biggest scores are selected. These are the query items that the Active user is asked to rate. We use this strategy as a baseline to verify that more 'intelligent' strategies can perform better than this.

- *Popularity* (POP): This strategy scores each candidate item by the total number of ratings for that item in K^{before}. The n_q items with the biggest scores are selected. This strategy is not personalized: if two users have the same pool of candidates C_u, then their Q_u will be the same.

- *Similarity-To-Profile* (S2P): Inspired by [RAC+02], we designed a personalized strategy that asks a user to rate items which are similar to ones she has already rated, under the assumption that such items are likely to be known by the user. Formally, S2P scores each candidate item by using $f_S(u, i, G = K^{before}) = \sum_{j \in I_u} \mathbb{1}(i, j)$, where $I_u = I_{K_u^{before}}$ are the items rated by u in K^{before}; the indicator function $\mathbb{1}(i, j)$ takes the value 1 if the item

i is one of the N_{S2P} nearest neighbors of the item j, i.e. $i \in \text{KNN}(j)$, 0 otherwise. The N_{S2P} neighbors of an item j are calculated using the cosine similarity of their ratings as the similarity metric. Once scores are calculated, the n_q items with the biggest scores are selected.

- *Highest-Predicted* (HP) [ERR14]: For every Active user $u \in U_{Active}$ and for each item in the candidate set of the user u, i.e. C_u, the Matrix Factorization recommender M (see next section) predicts the user's rating $\hat{r}_{u,i}$, so that $f_S(u, i, G = M) = \hat{r}_{u,i}$. Once scores are calculated, the n_q items with the biggest scores are selected. This strategy is personalized, choosing items which the recommender thinks the user will most like.

- *Binary-Predicted* (BP) [ERR14]: The matrix K^{before} is binarized to give a new matrix B, where $b_{u,i} \in B$ is 1 if the corresponding rating is available in K^{before}, i.e. $r_{u,i} \in \mathbb{R}$ and $b_{u,i}$ is 0 if the rating is not available, i.e. $r_{u,i} = \perp$. An implicit Matrix Factorization model M is built from B. For every Active user $u \in U_{Active}$ and for each item in the candidate set of the user u, the implicit model predicts the user's score $\hat{r}_{u,i}$, so that $f_S(u, i, G = M) = \hat{r}_{u,i}$. Once scores are calculated, the n_q items with the biggest scores are selected. The strategy is personalized, choosing items that are likely to be familiar to the user.

5.3.3 Methodology

Firstly, we randomly partition D into K^{before}, H^{he} and T^{he} where K^{before} is 60%, H^{he} is 20% and T^{he} is 20% of D.

Based on the discussion in Section 5.2, we can distinguish three evaluation methods, which differ depending on which sets are debiased. The three evaluation methods differ in how H and T are generated, as follows:

- *INT_HT*: In this method, we use WTD_H to intervene on T^{he} to generate an unbiased test set T. The size of T will be $\rho^T \times |T^{he}|$, where $\rho^T \in [0, 1]$ is a sampling rate. In this method, we also use WTD_H to intervene on H^{he} to generate an unbiased hidden set H. The size of H will be $\rho^H \times |H^{he}|$, similarly. We recall that this method aims to mitigate the bias in both the test and hidden sets.

- *INT_T*: This method aims to mitigate the bias in the test set only. In this method, we again use WTD_H to intervene on T^{he} to generate an unbiased test set T. However, in this method, as discussed above, we do not

use WTD_H to debias the hidden set, H^{he}. However, to ensure a fair comparison between *INT_HT* and *INT_T*, we need to make *INT_T*'s hidden set the same size as *INT_HT*'s. Otherwise, differences in the results of experiments might be due to *INT_T* having a larger hidden set. Hence, in this method, we randomly sample H from H^{he} using the same sampling rate ρ^H to produce a hidden set H.

- *CLASSIC*: This method corresponds to the traditional way of evaluating an AL strategy, where there is no attempt to mitigate the bias in the dataset (i.e. similarly to the one described in Section 5.1). To make comparisons between *CLASSIC*, *INT_HT* and *INT_T* fair, we randomly sample from T^{he} and H^{he} to get T and H using the same sampling rates as above, ρ^T and ρ^H, respectively.

To generate H from H^{he} and T from T^{he}, we set $\rho^H = \rho^T = 0.5$. We set this proportion to 0.5 based on results obtained in Section 3.4 where WTD_H was found to be a lower bound for effectively debiasing a test set. Of course, here we are using different datasets from those used in Section 3.4. The values of the sampling proportion to debias a test set we found previously for WBR3 and COAT may not carry over to the ML1M and LT datasets and, without any MAR data for ML1M and LT, we cannot find ρ^H and ρ^T in the way we did before. However, we pick $\rho^H = \rho^T = 0.5$ as a compromise that allows us to perform a substantial debiasing action on H^{he} and T^{he}; we argue that smaller values than 0.5 can result in intervened sets that are too small to give reliable results; larger values than 0.5 can mean that intervened sets are not appreciably different from the biased heldout sets. We also recall that, when using WTD_H above, we use formulas 3.13 and 3.14 but we must calculate different weights for each different intervention. For the MAR posteriors, we use the hypothesized distributions that we gave earlier (i.e. $P_{mar}(u|\mathcal{O}) = 1/|U| \quad \forall u \in U$ and $P_{mar}(i|\mathcal{O}) = 1/|I| \quad \forall i \in I$).

At this point, for each of the three evaluation methods, we have generated K^{before}, H and T. All three methods have the same (biased) training set K^{before}. *CLASSIC* and *INT_T* have the same (biased) H, but different T, i.e. biased for *CLASSIC*, debiased for *INT_T*. *INT_T* and *INT_HT* have the same (debiased) T, but different H, i.e. biased for *INT_T*, debiased for *INT_HT*. To complete the setup of our comprehensive evaluation, we must now choose a group of users to be Active users: we randomly select 30% of the users among the ones that have at least one rating in K^{before}. To the best of our knowledge, no best prac-

tice exists in the literature to fix this proportion. We chose this in order to have enough Active users and non-Active users to be able to show both user-centric and system-wide results. Also, we take K^{before} and, following the procedure described in 5.2.4, we obtain K^{val}, H^{val} and T^{val} with proportions 60%-20%-20% of $|K^{before}|$ respectively. We will use K^{val}, H^{val} and T^{val} to tune hyperparameters of the recommender and the AL strategies. Figure 5.2 represent graphically the three evaluation methods.

We evaluate each of the AL strategies described in Section 5.3.3 by means of an AL iteration. First, we must perform the *Before* step, i.e. training an RS on K^{before}. We use Matrix Factorization with a ranking loss function [PZT10].[3] This RS has two hyperparameters, i.e. the number of latent factors and a regularization term. We follow the procedure described in Section 5.2.4 to set them, where the former hyperparameter takes values in $V = \{20, 40, .., 200\}$ and the latter in $Z = \{0.001, 0.006, 0.01, 0.06, 0.1, 0.6\}$ and we optimize for Recall on the top-10 recommendations list of each user in K^{before}. Some AL strategies have hyperparameters too. S2P has N_{S2P}, and its value is selected from V following the procedure described in Section 5.2.4, again optimizing for Recall on the top-10 recommendation lists of the Active users in U_{Active}. Hyperparameters for HP do not need to be tuned, because the recommender model is used to score the candidates. BP also uses a factorization model, where the number of latent factors (chosen from V) and a regularization term (chosen from Z) do need to be tuned. Therefore, we tune them in the same way we did for the HP recommender but instead optimizing them for Recall on the top-10 recommendation lists of all users in U, where all items in T^{val} are relevant items.

To allow a fair comparison between AL strategies and across the three evaluation methods, this *Before* step is performed only once. Then, each AL strategy performs its *AL interaction* step, where it produces a query composed of $n_q = 50$ items for each Active user, for a total AL budget of $50|U_{Active}|$ rating requests. Of course, in practice it is unlikely that an RS would ask a user for 50 ratings. Our choice of $n_q = 50$ is experimentally motivated by the fact that we need to elicit enough ratings for there to be an appreciable change in the performance of the RS.

Testing the quality of the recommender is the same both *Before* and *After* the new ratings are acquired. For each method, we compute Recall, EILD, EPC and ECBS for top-10 recommendations on T, for each user in U_{Active} (see formulas

[3] We use the implementation from the RankSys library: https://github.com/RankSys

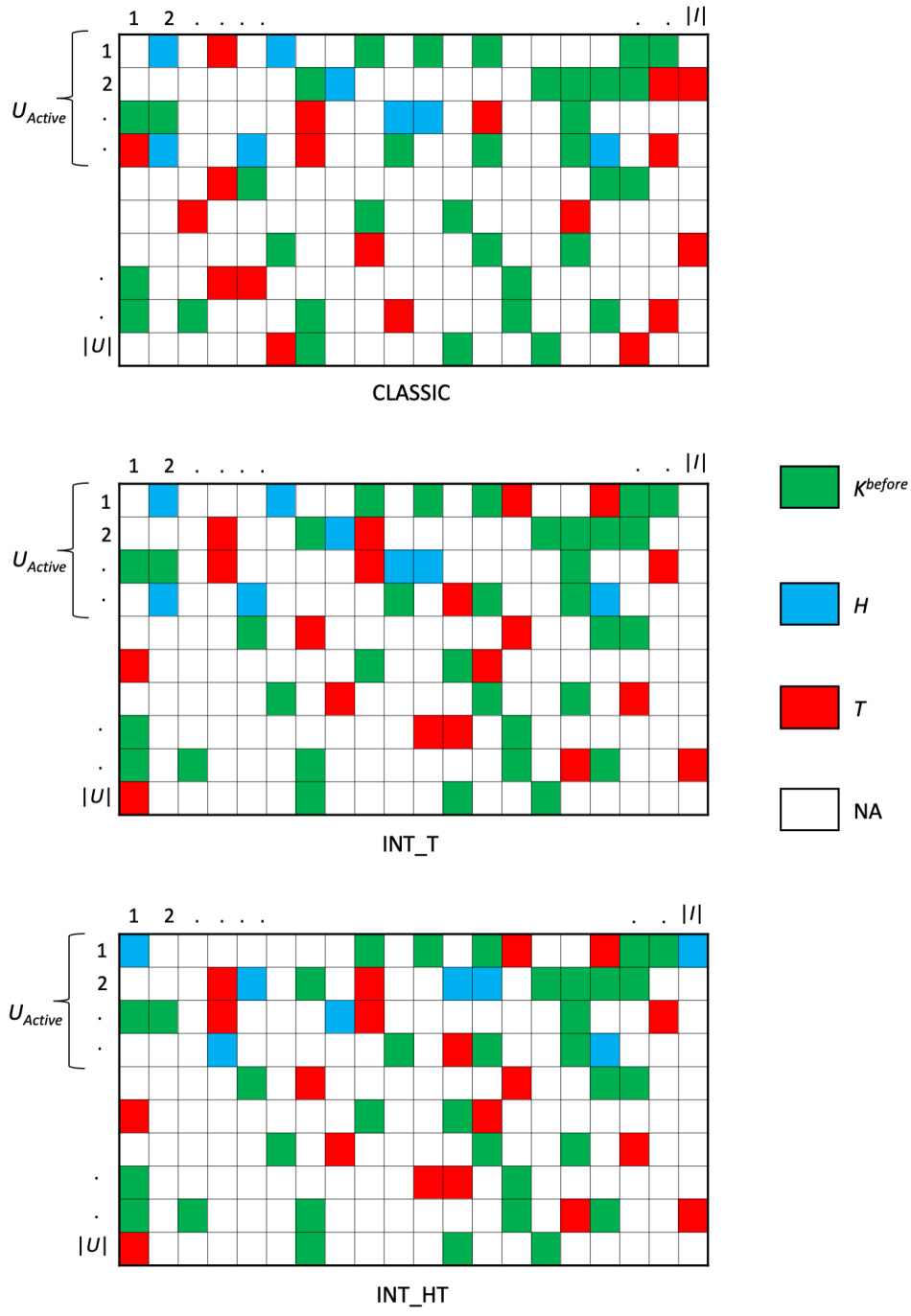

Figure 5.2: Visualization of the dataset split for *CLASSIC*, *INT_T* and *INT_HT* methods. NA stands for Not Available for the experiments, i.e. a rating that is either not available in D or discarded when debiasing the data. Also, in all three methods, U_{Active} are randomly-selected from U. The figures simplify by representing them as if they occupy the first rows of the matrix.

Table 5.2: ML1M. The table reports, for each strategy and for each evaluation method, the average number of elicited ratings, the average number of elicited ratings per Active user and the average number of users who are Respondents. Averages are calculated across the 10 folds.

	CLASSIC / INT_T			INT_HT		
	# elicited	# elicited per user	# Resp.	# elicited	# elicited per user	# Resp.
RND	304.2	0.21	260.8	302.7	0.21	263.7
POP	3428.6	2.34	1166.3	512.5	0.35	426.4
S2P	5108.7	3.48	1323.3	3947.0	2.69	1290
HP	5995.2	4.09	1372	3107.6	2.12	1241.7
BP	6170.7	4.21	1376.5	3140.8	2.14	1237

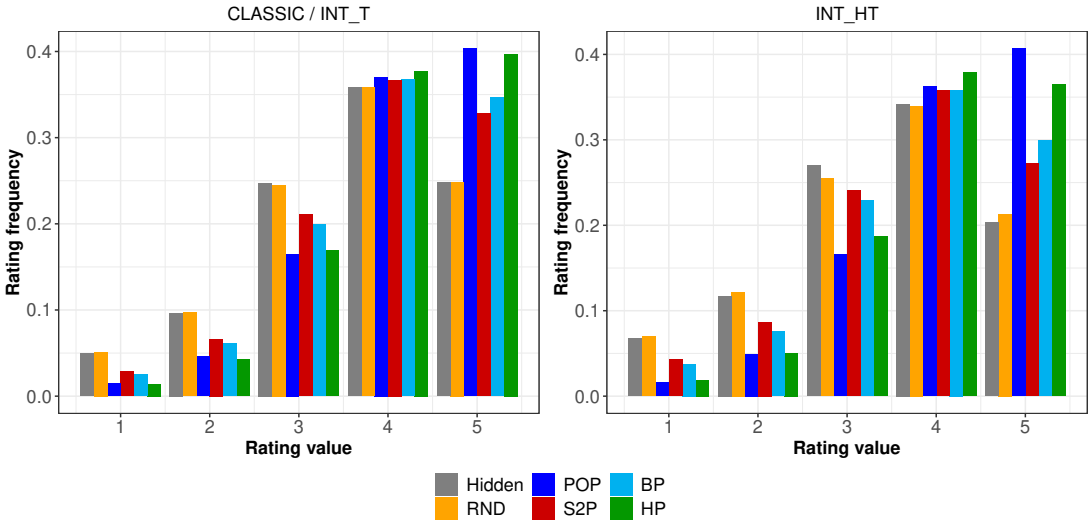

Figure 5.3: ML1M. Values distribution for the elicited ratings by different AL strategies.

2.2, 2.6, 2.9, 2.12 in Section 2.1 respectively). EILD, EPC and ECBS need a pairwise item distance metric $\mathrm{dist}(i, j)$ to be defined. In this case study, we use the Jaccard distance between the features of items i and j. As item features, we use the movies' genres in the case of ML1M and tags in the case of LT.

There is, of course, the usual danger that the results obtained are specific to this particular split of the data. Hence, we re-partition D and repeat the AL iteration 10 times, and report the average results over these 10 runs.

5.3.4 Results for ML1M

Table 5.2 and Figure 5.3 help to run a preliminary analysis on ML1M results. Table 5.2 shows that, as expected, different strategies elicit different numbers of

ratings. To note, *CLASSIC* and *INT_T* have the same values because they share the same K^{before} and H, hence the same sets of ratings get elicited; *CLASSIC* and *INT_T* differ only in their test sets, which affects the performance results that we show later. As we can see, the RND AL strategy is unaffected by whether we debias the hidden set (*INT_HT*) or not (*CLASSIC*, *INT_T*), and it is able to elicit very few ratings from roughly only a quarter of the Active users. But the 'intelligent' strategies (POP, S2P, HP, BP) are affected. These strategies elicit more ratings on average when the hidden set is not debiased (*CLASSIC*, *INT_T*) than when the hidden set is debiased (*INT_HT*). But, interestingly, the number of users from whom ratings are elicited is close in *CLASSIC/INT_T* and *INT_HT* (except for POP). Moreover, looking just at the results for debiased hidden sets (*INT_HT*), we see that S2P, HP, BP are the strategies which elicit the largest number of ratings; and we have that POP elicits a slightly larger number of ratings than RND. We would expect this too, if we have successfully removed popularity bias from the hidden sets. Lastly, HP and BP have very similar results both in *CLASSIC/INT_T* and *INT_HT* but we observe they elicit more ratings than S2P according to *CLASSIC/INT_T*, while it is the opposite for *INT_HT*.

Figure 5.3 tells us how rating values elicited by different strategies are distributed. Both *CLASSIC/INT_T* and *INT_HT* show similar behaviours for all the strategies. As expected, RND elicits ratings which are distributed very similarly to the ones available in the hidden sets (where the distribution is slightly skewed towards relevant ratings, i.e. roughly, relevant ratings account for the 60% of the total number of hidden ratings in *CLASSIC/INT_T*, and 55% in *INT_HT*). 'Intelligent' strategies increase this skew, eliciting many more relevant ratings than non-relevant ones (especially POP and HP, which is to be expected).

Results for Active users and system-wide perspective

Now, we analyse the impact of AL on the recommender's performances. The heat map in Figure 5.4 reports results on two groups of users, i.e. Active users and all users in U, i.e. the latter being the system-wide perspective. The statistical significance of the results is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine whether the recommender performance before the AL iteration is statistically different from its performance after the strategy has elicited some ratings. The second set of tests are pairwise comparison tests between AL strategies, to determine

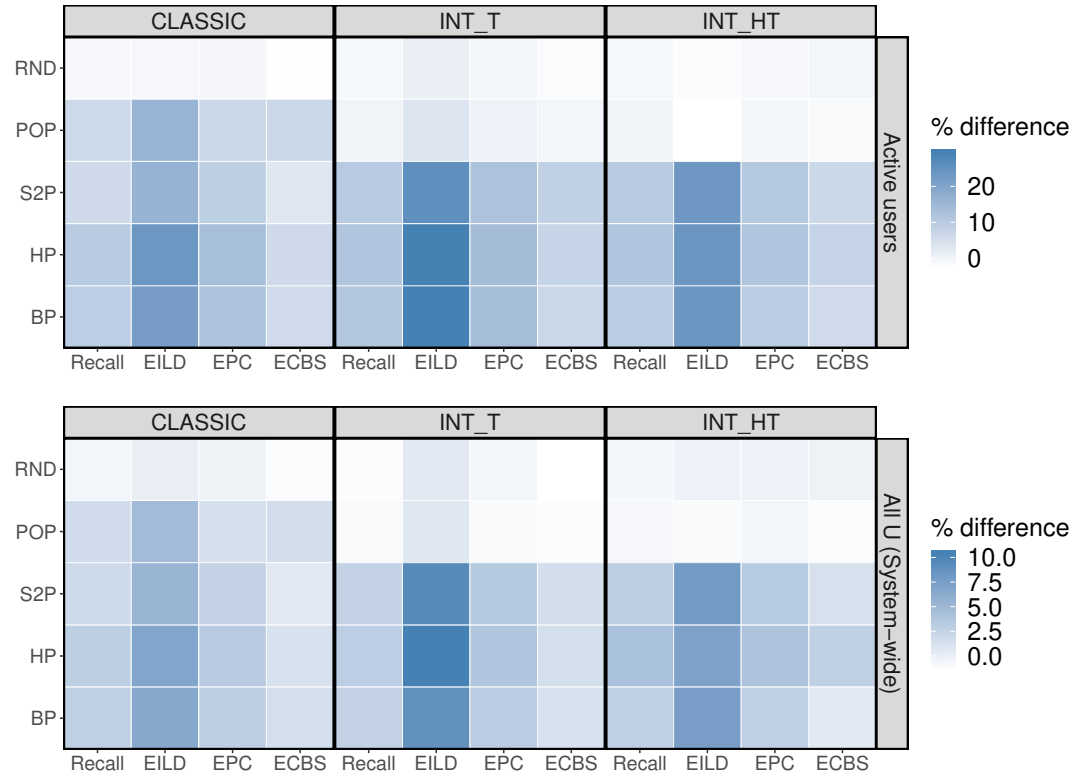

Figure 5.4: ML1M. The heat maps for ML1M show results on two groups of users, i.e. Active users and U. For each different strategy and for each different metric, values are expressed in terms of percentage difference between the recommender's performance before applying the AL iteration and after applying the AL iteration.

whether the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy. All these tests are performed using the two-tailed Wilcoxon signed rank tests presented earlier and results are available in Tables A.4 and A.5.

The first observation is that AL helps an RS improve its recommendation quality: this is demonstrated by all three evaluation methods where the heat map shows positive percentage differences across all metrics and for both groups of users. As expected, percentage differences are generally higher for Active users than when considering results from all the users in the system. (Later in this section, we will also analyse the impact of AL for Respondents and non-Respondents only.)

In detail, for Active users and for all U, RND brings a non statistically significant difference to the recommender's performance, according to all three evaluation

methods. Table 5.2 shows indeed that asking users about random items elicits very few ratings on average and the new ratings seem to make no difference for the system. Results for the other strategies are as follows:

- For Active users under the *CLASSIC* method, HP and BP show the best improvement over the other strategies and they are really similar in this setting (their performances are not statistically significantly different from each other except for EILD and EPC). Performances of POP and S2P are worse than HP/BP, better than RND, and similar to each other (not statistically significantly different from each other except for EPC and ECBS).
- Things change when we look at Active users under *INT_T* and *INT_HT*. In particular, POP's performance is now low and similar to RND's (indeed they are not statistically significantly different from each other for except for ECBS). POP's performance on *INT_T* (where H is biased) can be explained by the fact that the strategy acquires a good number of ratings for popular items (because of H's bias), which might bias the RS towards popular recommendations; such a recommender will perform poorly when evaluated on a debiased test set. On *INT_HT* (where H is debiased), POP can acquire only a few more ratings than RND; consequently, we argue that this is not enough to make its performance stand out from RND. S2P, HP and BP are the best strategies under both *INT_T* and *INT_HT* and they are similar to each other (indeed, their performances are not statistically different to each other for most of the metrics).
- Similar conclusions to the ones above apply for the U group (the system-wide results) according to all three evaluation methods, so that full analysis can be omitted.

To sum up our analysis for Active users and the system-wide perspective according to all three evaluation methods, the baseline RND does not bring any benefit to the recommender, as expected. Regarding the other four strategies, the traditional evaluation and our debiased evaluation reveal different findings. While HP and BP are the best strategies for both methods, POP and S2P are good strategies for *CLASSIC*, while the debiased evaluation uplifts S2P to be one of the best strategies and significantly lowers POP to be comparable to RND.

Results for Respondents

This section focuses our lens on those users who provide at least one rating during the AL iteration, i.e. the Respondents. Also, from now on we will consider only results when debiasing both the hidden and the test sets, i.e. *INT_HT*, which we believe is the most reliable method to evaluate the impact of AL. (However, results on *INT_T* are similar.) Additionally, we exclude from our analysis the RND strategy, which was shown earlier to be ineffective at impacting the recommender's performance (and therefore not really interesting to us).

For Respondent users, in Figure 5.5 we plot the Recall percentage difference values against the number of ratings elicited per user (subplot 5.5.a) and against different beyond-accuracy measurements (subplots 5.5.b, 5.5.c, 5.5.d). Subplot 5.5.a shows to what extent the number of elicited ratings impacts the Recall performances of the recommender, across different strategies. We can see there is a (rough) linear correlation between the two measurements, such that the more ratings elicited, the better the Recall improvement. This is expected to some extent: while POP elicits very few ratings per user (bringing a small and not statistically significant improvement to Recall), S2P, HP and BP elicit many more ratings, impacting the Recall improvement by more than 10%. Although S2P, HP and BP display Recall improvements which are close to each other, interestingly HP obtains the highest value despite eliciting the fewest ratings per user of these three strategies. This suggests that HP's elicited ratings are *qualitatively* better than the ones elicited by the other strategies. HP seems to be a better strategy also according to the other three subplots. In other words, it is also the best at improving diversity (EILD), novelty (EPC) and surprise (ECBS).

Table 5.3 and Figures 5.6 and 5.7 can help to figure out the reason why AL strategies have different performances. In particular, Table 5.3 shows how different are the top-10 recommendations lists provided to Respondents after the AL step (with respect to the ones provided to them before the AL step). As we can see, the recommendations provided after the strategies have elicited some ratings contain between 12% and 19% of items that were not included before. It is quite surprising that recommendations after HP change by a smaller extent (12%) than the ones provided by BP and S2P, regardless of the fact that HP gets the best improvement over all metrics (Figure 5.5). A slightly different but related outcome is found for POP: recommendations produced afterwards for those who provided some ratings are quite different (17%) but performance change is quite small. All this confirms that it is both the amount and the quality

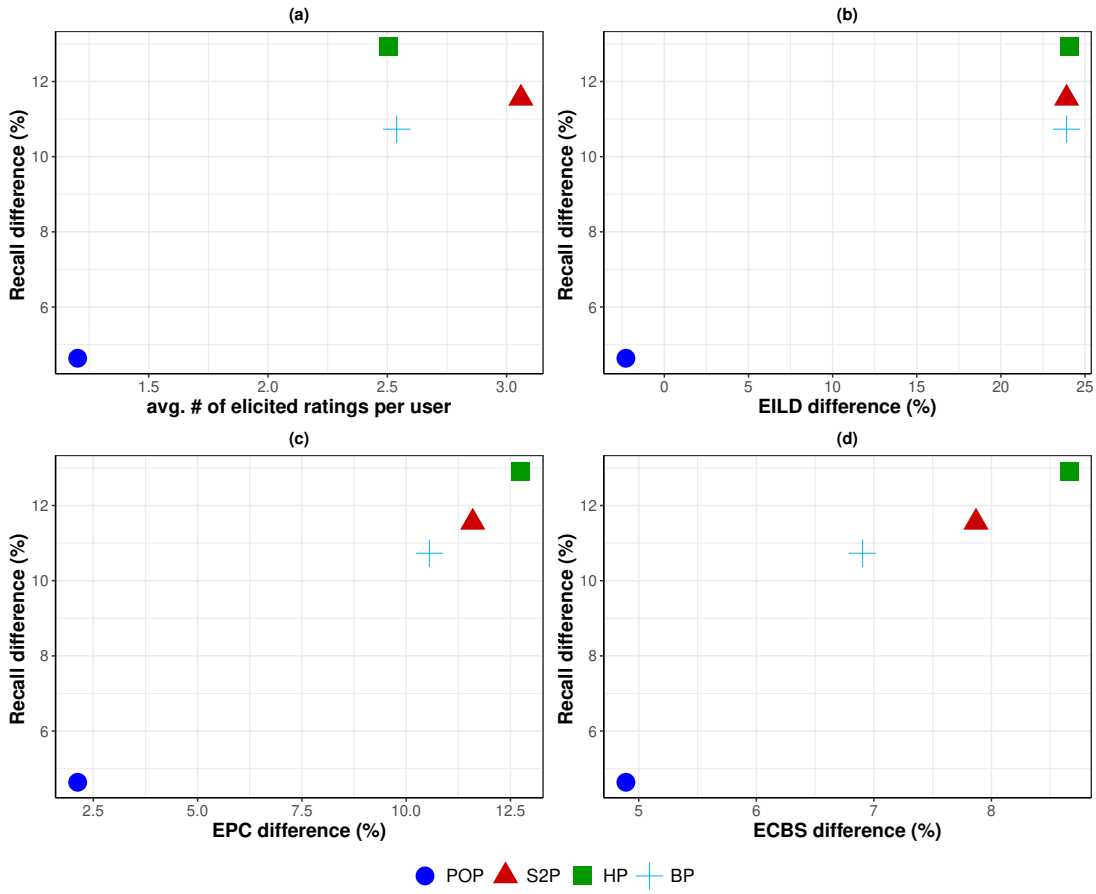

Figure 5.5: ML1M. Respondent users results for the *INT_HT* evaluation method. All four subplots display the Recall percentage difference values on the y-axis. On the x-axis, we display the average number of elicited ratings per user in subplot a; we display EILD, EPC and ECBS percentage difference values in subplots b, c and d respectively.

of the ratings elicited that impact the recommendations provided by the recommender (and consequently performances too). New ratings from POP are able to change the recommendations quite significantly but with a limited benefit for the users; on the other hand, new ratings from HP and BP change recommendations less than the ratings from POP, but the changes have a much greater benefit for the users. S2P is between the two. For comparison and completeness, Table 5.5 also includes the same statistics for users that did not provide any new ratings. Those results will be discussed in the next section.

Figures 5.6 and 5.7 drill into the results by analysing user-item pairs that are selected by each strategy for composing the queries for the Active users (Figure 5.6) and by analysing user-item pairs where a rating is successfully elicited by each strategy among the ones available in the hidden set H (Figure 5.7).

Table 5.3: ML1M. Average percentage difference of the top-10 recommendation lists for Respondent users provided by the recommender after the AL step with respect to the ones provided before the AL step. To note, we chose Recall because increasing accuracy is usually considered the primary objective of a recommender system.

	Respondents	*U-less-Respondents*
POP	17%	14%
S2P	19%	17%
HP	12%	6%
BP	13%	8%

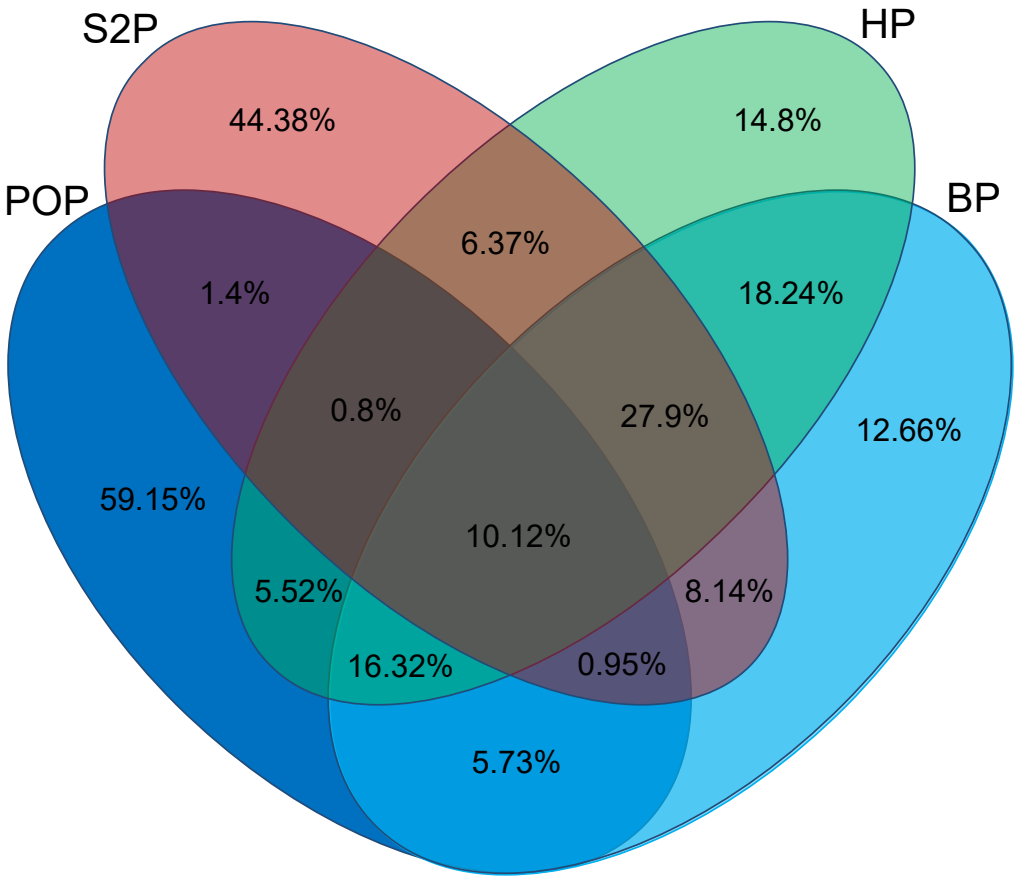

Figure 5.6: ML1M. Overlaps between queries made to Active users by different AL strategies. Proportions are calculated with respect to the total number of ratings requested by each strategy.

The Venn diagram in Figure 5.6 shows that roughly 10% of user-item pairs are selected by all strategies, which is a not negligible percentage. This means that these strategies agree on the usefulness of considerable number of the candidates, if acquired. At the same time, fewer than 1% of the candidates on

which they agree result in ratings being elicited, i.e. a quite small percentage compared with the other ones in the Venn diagram of Figure 5.7.

Something emerging from Figure 5.6 is that each strategy selects its own set of 'unique' questions for the users, and percentage values for these are among the highest. We can observe a similar trend in Figure 5.7, where, for each of the considered strategies, the majority of its elicited ratings are exclusively elicited by only one strategy (except for POP, that can elicit less than 1% of the hidden ratings in any case). This suggests that strategies are substantially different from each other, in the first place because of what they ask users to rate, and second (and, to some extent, consequently) because of what they elicit. An exception to this finding is the pair HP and BP. The diagrams show that: they ask similar questions to Active users (roughly 70%) and their intersection is much bigger than all the other pairwise intersections; and they also share many elicited ratings (roughly 12%). These similarities between HP and BP can explain why their performances are close to each other more than to any other strategies. To conclude the analysis, we can observe from Figure 5.7 that the majority of the hidden ratings (roughly 74% of them) have not been elicited by any strategy. This raises a further, and still unanswered, question: what would the performances have been like if those ratings were elicited? In other words, how useful would they have been at improving the recommender's performance? In Section 6.1.2 we will address those questions when evaluating new AL strategies that we design.

Results for U-less-Respondents

One of our objectives in designing our more comprehensive evaluation framework was to be able to measure the impact of AL on users who do not provide any ratings. Here, we report results for *INT_HT* only (like before). However, in this case study, only a few performance values are statistically significant: the ones of POP (-1% percentage difference on Recall and -1.4% percentage difference ECBS) and one of HP (+1.1 percentage difference on EPC). Thus, we claim that such evidences are not strong enough to consider any strategies to be really effective on helping those type of users.

Results by profile-size

In this section, we show results by profile size. This enables us to compare the impact of AL on cold-start users and more mature users, as it is one of our

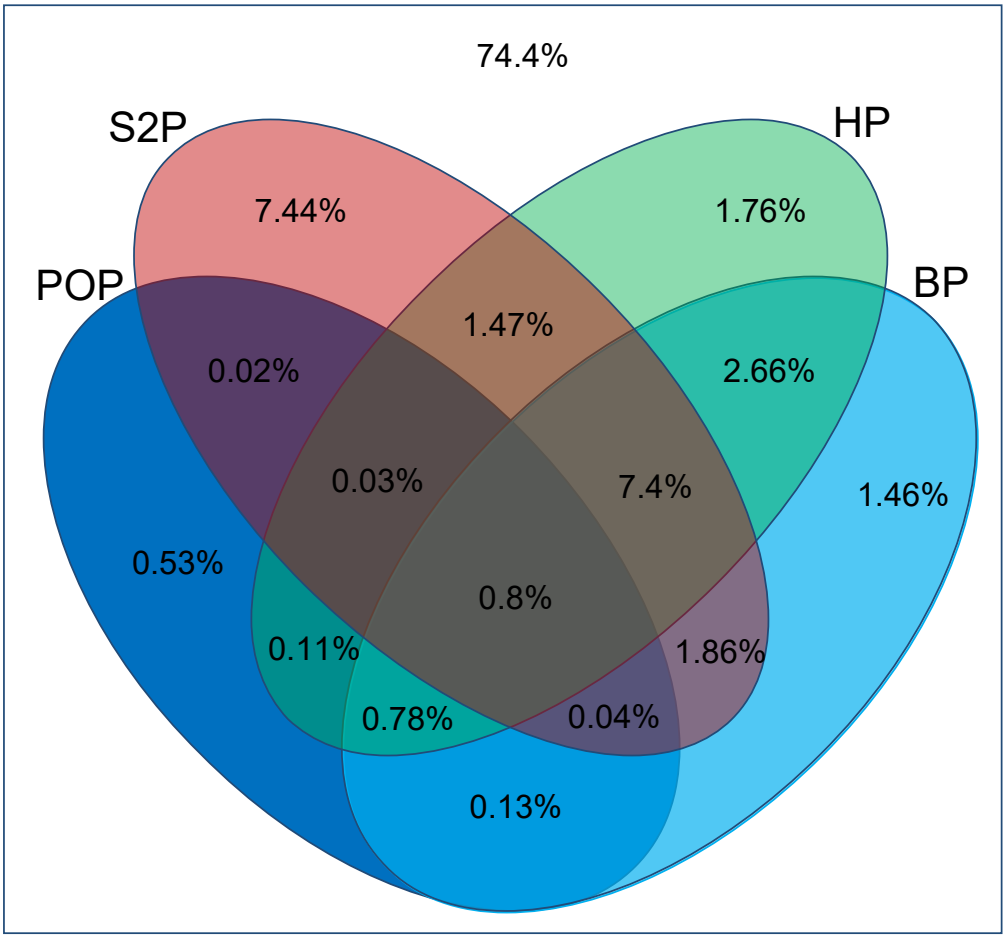

Figure 5.7: ML1M. Overlaps between elicited ratings of different AL strategies. Proportions are calculated with respect to the total number of hidden ratings belonging to the Active users, i.e. such ratings that could have been elicited.

objectives for this evaluation. To do this, we put each Active user u into a bucket based on her initial profile size $|K_u^{before}|$, i.e. the one available in the training set before running the AL iteration. The design of the buckets in this case study is somewhat heuristic. The first bucket comprises user profiles that contain 1-20 ratings: we define such users to be cold-start users. This is inspired by the literature (e.g. [KK14, FTBE+16]), where a user is labelled as a cold-start user if her profile is in this specific range. We define mature users to be ones associated with profiles that contain more than 20 ratings and the buckets which contain such users are chosen in a way that ensures that each bucket contains at least 50 users. After this bucketing procedure, users are divided into seven buckets, with ranges 1-20, 21-50, 51-100, 101-150, 151-200, 201-300, >300. Also, statistical significance tests for each single bucket are performed using the two-

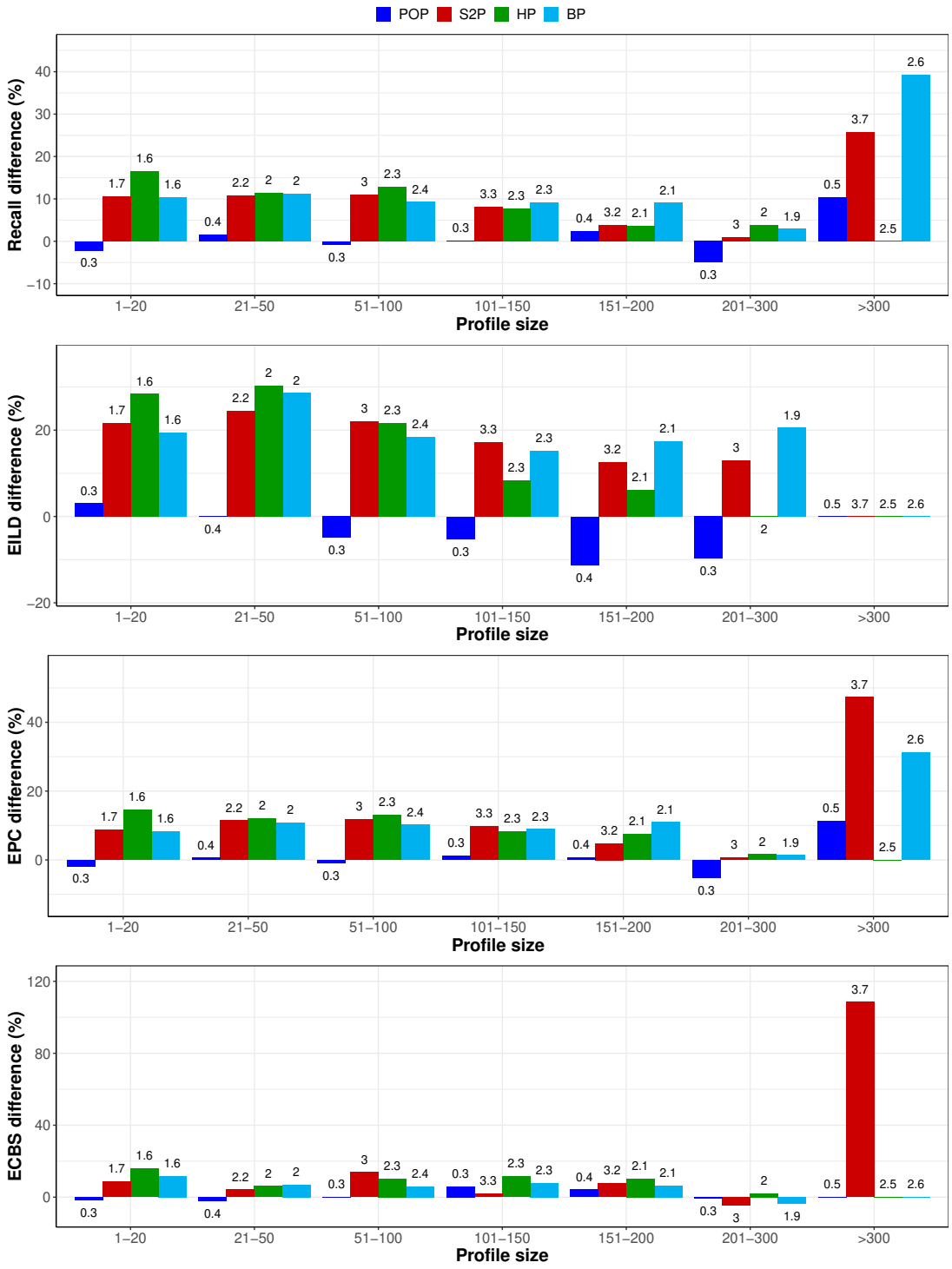

Figure 5.8: ML1M. Active user results by profile size. On the top of each bar, we report the average number of ratings elicited by the corresponding strategy.

tailed Wilcoxon signed rank tests presented earlier and results are available in Table A.6.

Figure 5.8 shows the buckets and their results. First, we can observe that S2P,

Table 5.4: LT. The table reports, for each strategy and for each evaluation methods, the average number of elicited ratings, the average number of elicited ratings per Active user and the average number of users who are Respondents. Averages are calculated across the 10 folds.

	CLASSIC / INT_T			INT_HT		
	# elicited	# elicited per user	# Resp.	# elicited	# elicited per user	# Resp.
RND	24.6	0.01	23.8	24.0	0.01	23.8
POP	1303.8	0.73	767.1	7.3	0.004	7.3
S2P	2701.8	1.51	1171.1	1202.3	0.67	591.3
HP	3464.1	1.94	1328.3	1156.4	0.65	679.4
BP	3308.7	1.85	1276.5	896.7	0.50	548.7

HP and BP are effective strategies across all buckets and all metrics, while POP again is poor. (Indeed, we will leave out POP from the rest of the analysis).

We can observe that, for cold-start users, despite the fact that HP elicits roughly the same amount of ratings as S2P and BP, it is the best strategy according to all metrics. However, statistical tests on such results strongly limit this finding. HP's performances are: not statistically significant for EILD; not significantly different from the ones of S2P; and significantly different only from BP on Recall and EPC.

For mature users, in the first bucket (i.e. 21-50) HP is still superior to S2P and BP. In the rest of the buckets, it is not clear which of the strategies is the best one (apart from being able to elicit a different number of ratings), because none of the three consistently outperforms the others, i.e. we cannot infer a clear pattern along buckets and across the metrics. This is confirmed by the statistical significance tests performed, where performances of S2P, HP and BP are not statistically different to each other most of the time. However, these findings are somehow expected, considering the results analysed earlier for Active users (Figure 5.4).

5.3.5 Results for LT

In this section, we describe results for LT. However, this report will be shorter due to the similarity of the results with the ones of ML1M.

In Table 5.4, we show the statistics of the elicitation process. The situation is quite similar to the one in Table 5.2 for ML1M. RND elicits very few ratings, while POP elicits a good number of ratings in the *CLASSIC* and *INT_T* methods

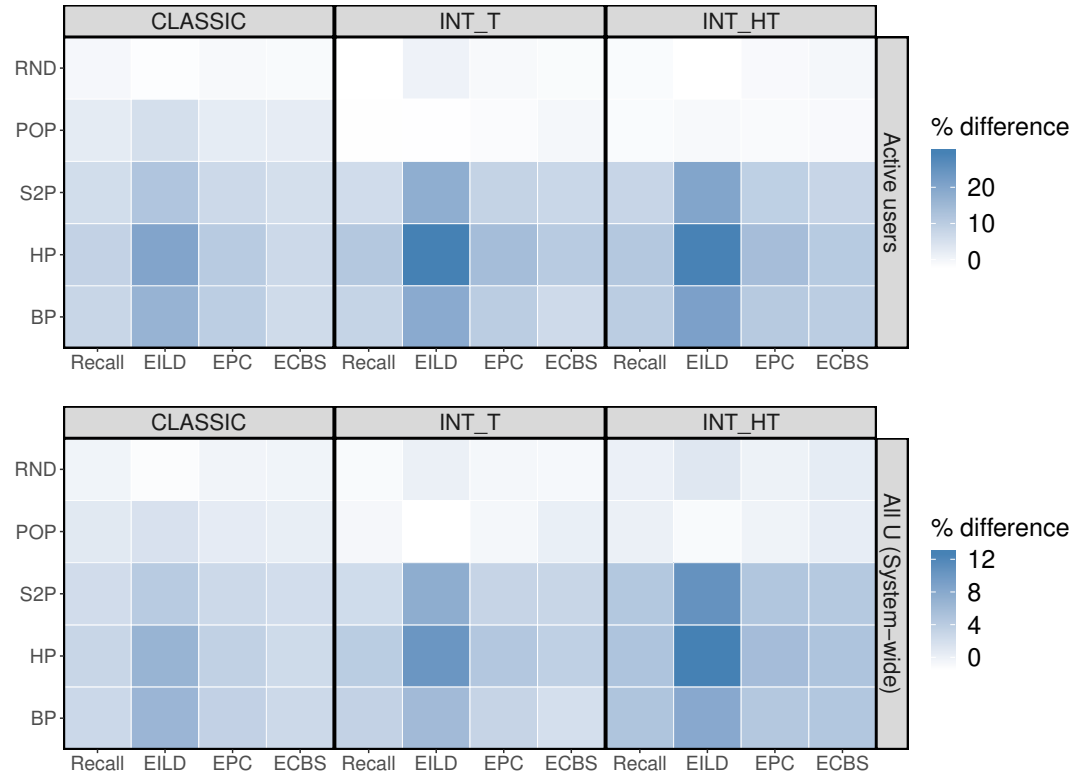

Figure 5.9: LT. The heat maps for LT show results on two groups of users, i.e. Active users and U. For each different strategy and for each different metric, values are expressed in terms of percentage difference between the recommender's performance before applying the AL iteration and after applying the AL iteration.

but not the *INT_HT* method (where it is even worse than RND). S2P, HP and BP are much superior, with HP being the best at acquiring ratings from many users according to *CLASSIC/INT_T*. An interesting finding also is that, according to *INT_HT*, S2P acquires the greatest number of ratings but from fewer users than HP, i.e. the number of Respondents for HP is bigger than for S2P. BP is between HP and S2P on *CLASSIC/INT_T* and the worst of the two on *INT_HT*.

Results for Active users and system-wide

Figure 5.9 for LT is very similar to Figure 5.4 for ML1M and both Active users and system-wide results follow the same trend (and the statistical significance tests performed on those results are reported in Tables A.7 and A.8.). RND is the worst strategy, and brings a not statistically significant improvement to the recommender for all metrics. According to *CLASSIC*, HP and BP are the best strategies (and their performances are not statistically different to each other except for EILD); POP and S2P are inferior to HP/BP, with S2P being

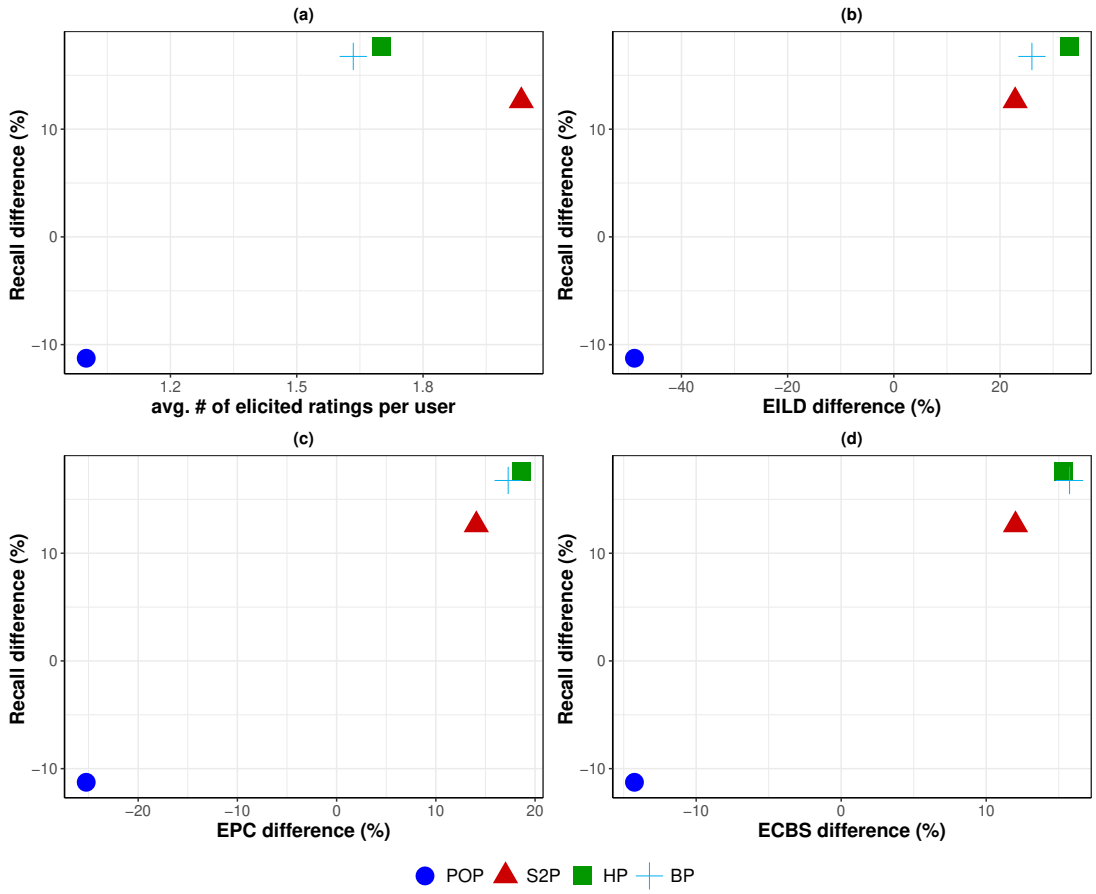

Figure 5.10: LT. Respondent users results for the *INT_HT* evaluation method. All four subplots display the Recall percentage difference values on the y-axis. On the x-axis we display the average number of elicited ratings per user in subplot a; we display EILD, EPC and ECBS percentage difference values in subplots b, c and d respectively.

better than POP. This changes for *INT_T* and *INT_HT*: POP does not bring a statistically significant improvement to the recommender for all metrics; HP is definitely the best strategy, while S2P and BP are inferior to HP and similar to each other (and their performance is most of the time non statistically different to each other across the various metrics).

Results for Respondents

Results for LT Respondents in Figure 5.10 are similar to the ML1M results in Figure 5.5. According to the *INT_HT* method, HP brings the most benefit to the users who provide ratings (even though BP is very close to it on EPC and ECBS). However, differently from the ML1M results, POP percentage differences are all negatives for every metric considered, i.e. POP significantly harms

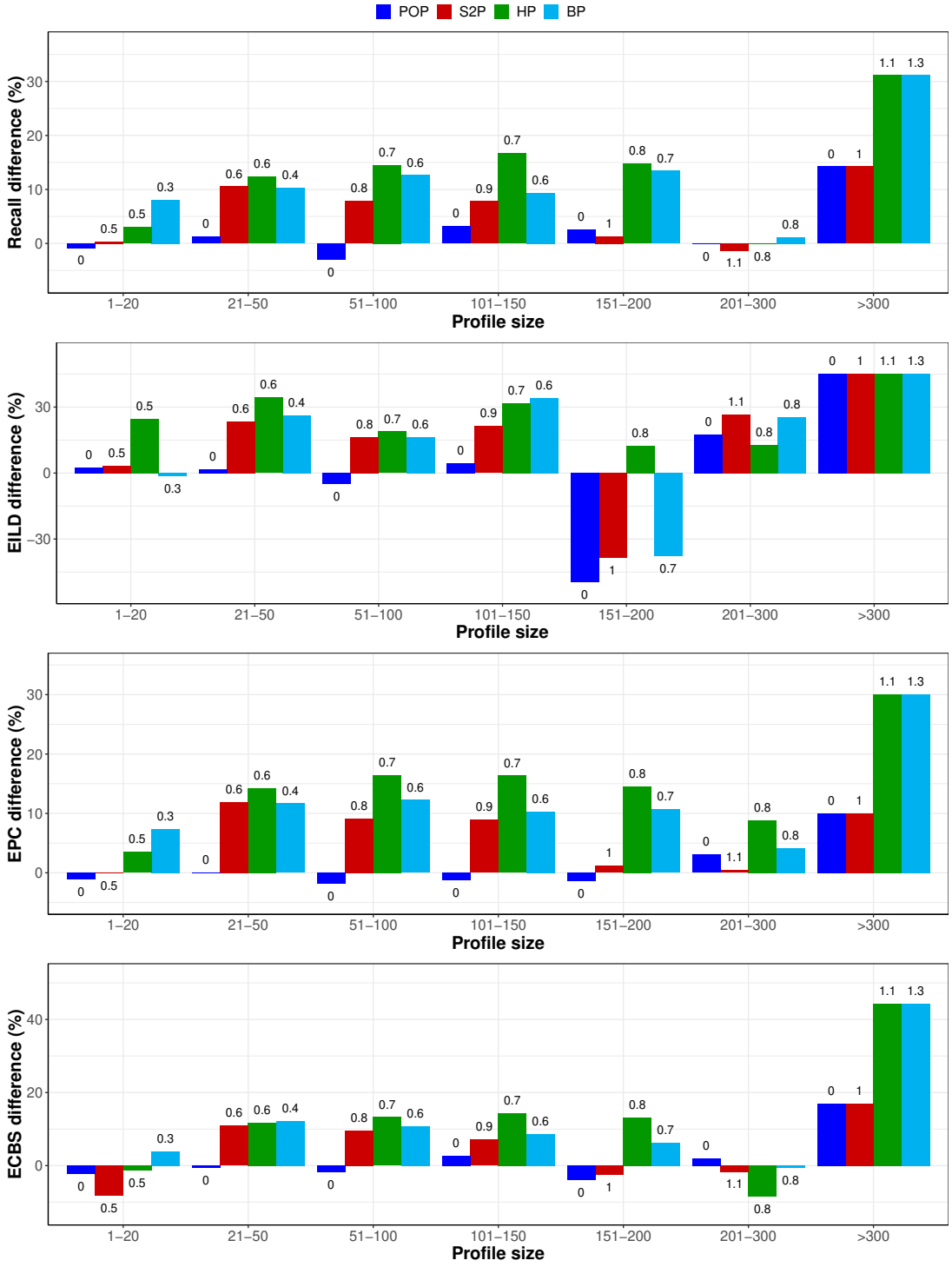

Figure 5.11: LT. Active users results by profile size. On the top of each bar, we report the average number of ratings elicited by the corresponding strategy.

recommendations. Note, however, that POP's performances are from just seven Respondent users, which might not be representative enough to draw meaningful conclusions.

Results by profile-size

Finally, LT results by profile size in Figure 5.11 are also similar to the ones for ML1M. A difference is in the bucket of the cold-start users, where BP seems to be the most effective strategy across all the metrics, except for EILD. However, statistical significance tests reported in Table A.9 show that results for this bucket are not significant. For mature users, HP is consistently superior to S2P and BP up to the 151-200 bucket, according to all metrics; for subsequent buckets, it is not clear which strategy is the best. However, statistical significance tests show once again that performances of S2P, HP and BP are not significantly different to each other most of the time.

5.4 Conclusions

In this chapter, we proposed a new offline experimental methodology to evaluate the effectiveness of AL strategies for RSs. One goal of the evaluation methodology is to mitigate the bias introduced by the use of an MNAR observed dataset, which is a problem not considered by the traditional evaluation methodology widely used in the literature. We presented two alternative methods for conducting such a debiased evaluation: one debiases only the test set; the other debiases both the test set and the hidden set. Another goal of our evaluation methodology is to assess an AL strategy's performance more comprehensively than the classic evaluation: our framework allows to evaluate an AL strategy by considering its impact under different perspectives, e.g. on mature users and cold-start users; on users who provide new data, who do not provide any new data and all the users in the system; and on beyond-accuracy qualities of recommendations provided to users.

Using both classic and debiased evaluations, our case study compares five simple AL strategies from the literature on two widely-used MNAR datasets. Those strategies are: RND, which asks the users to rate random items; POP, which asks the users to rate the most popular items in the system; S2P, which asks a user to rate items similar to the ones in her profile; HP, which asks users to rate items that the recommender thinks they will like; and BP, which asks users to rate items that most likely are familiar to them.

Results on both datasets show similar outcomes. In general, the best strategies are the ones that can elicit lots of ratings, i.e. there is a positive correlation between the amount of ratings elicited by a strategy and the consequent im-

provement of the recommender. But, we also found that the traditional biased evaluation shows different outcomes from the debiased one. In particular, according to our debiased methodology, performance of POP and RND are very similar, while on the classic method POP was unfairly uplifted with respect to RND. Regarding the other three strategies, experiments show that asking users to rate items selected intelligently has a positive impact on the systems. However, if the classic evaluation elects BP and HP as the best strategies for users who are queried and for the system-wide perspective, our debiased evaluation shows that also S2P is very close to the other two. (Indeed, most of their performances in terms of accuracy and beyond-accuracy measurements are not statistically different from each other across all metrics employed).

We also found that new ratings from Respondents do not have a significant impact on the recommendations provided to non-Respondents, irrespective of the strategy used. Despite this, we argue that such an analysis is still important to properly evaluate an AL strategy because our findings might be limited to the specific recommender model that we used, i.e. Matrix Factorization. Indeed, recommendations provided by a different recommender model to non-Respondent users might be impacted to a greater extent by those new ratings.

One decision to be made is whether to debias both the hidden set and test set, or just the test set. Our initial results revealed similar insights for both methods. Hence, it remains unclear whether one should be preferred over the other. However, we argue that it is probably more meaningful to evaluate a strategy in a setting where the strategy is given the chance to exploit as little bias as possible, i.e. when both the hidden set and the test set are debiased. Therefore, we performed the remainder of our analysis in the setting where both the hidden and test sets were debiased. We presented results for Respondent users, where HP is shown to be the best strategy according to all metrics (although S2P and BP are both close to it). Also, we analysed results by profile size, but finding little that was new: the three best strategies are still S2P, HP and BP and their performances are very close to each other, regardless of the size of a user's profile. HP and BP appear to be weakly superior for cold-start users on ML1M and LT respectively.

In the light of these findings, this suggests to practitioners the need to reconsider results presented in the literature by using instead a debiased evaluation method, as we did in this chapter. In the next chapter, we will focus on the design of new AL strategies and we will use our debiased methodology *INT_HT*

to evaluate them.

Chapter 6

Active Learning Beyond-Accuracy

The main goal of the AL strategies that have been proposed in the literature so far is to improve the accuracy of the recommender at hand. In this chapter, we design new AL strategies, but this time the focus will be on beyond-accuracy qualities. However, this does not mean that accuracy will be ignored or penalised: we argue that an AL strategy has to preserve accuracy as much as possible, along with increasing other beyond-accuracy qualities.

In the previous chapter, we proposed a new debiased evaluation methodology, i.e. *INT_HT*, that allows researchers to evaluate the effectiveness of an AL strategy at improving the quality of a recommender system. In this chapter, we use the methodology to evaluate our new AL strategies. Again using the ML1M and LT datasets, we compare the new strategies with HP, the best strategy accordingly to the results obtained in the case study of Section 5.3.

In Section 6.1 we will provide useful tools that will help us in designing new strategies. In Sections 6.2, 6.3 and 6.4, we propose new strategies targeted at improving diversity, novelty and serendipity respectively. In Section 6.5 we propose hybridization approaches to AL, and in Section 6.6 we wrap up with conclusions for this chapter.

6.1 On Designing New Strategies

Designing and evaluating a new AL strategy that is effective is challenging. There is the initial challenge of designing a function f_S for scoring candidate items (Algorithm 1) that encapsulates whatever approach is being considered, e.g. error reduction, uncertainty sampling, etc. But additionally, the design

must take into account the time complexity and practical efficiency of f_S, which affect both how easy it will be to evaluate the strategy and its subsequent use in a deployed system. Once a particular f_S is defined, another challenge is in the evaluation of the strategy, both offline and online. For example, offline experiments such as the ones we have performed in the case study of Section 5.3 require substantial computational costs to complete even for just a few strategies. Moreover, as we emphasised in the previous chapter, designers are typically required to evaluate many strategies in offline experiments before finding the ones that are promising and can be moved into a subsequent (and likely more expensive) online evaluation.

For these reasons, in this section we propose two additional tools which might help when designing and evaluating new AL strategies. In Section 6.1.1, we conduct an investigation that gives us insights into the relationship between the characteristics of users' profiles and performances of the recommender; this will be useful when designing a strategy for a specific beyond-accuracy objective. Then, in Section 6.1.2, we describe the usefulness of performing the evaluation of an AL strategy under an artificial setting (introduced earlier in Section 5.1.2) for both filtering out the weaker strategies and speeding up the whole evaluation. Using this setting, we also discuss an upper bound on the performances achievable by an 'ideal' AL strategy.

6.1.1 On the relation between user profiles and performance

In Active Learning, when designing a new strategy, we must bear in mind that the final goal is to improve the performance of the model at hand. In a recommender system scenario, besides Active Learning, we argued that the quality of the recommendations depends on the quality of the user profiles (see the discussion in Section 4.3). In other words, there is a correlation between the quality of a user profile and the recommendations provided to the user. For this reason, an AL strategy that improves the quality of a user profile is likely to (indirectly) benefit the recommender's performance as well. Defining the recommender's quality as the quality of its recommendations (e.g. a recommendation that is relevant and serendipitous to a user is a good recommendation) is widely accepted by practitioners. However, defining the *quality of a user profile* is not so straightforward. What does a 'good' user profile look like? How do we measure the quality of a user's profile? To the best of our knowledge, these questions are still unanswered in the literature. In this section, we attempt to

provide a partial answer to them, by first proposing a few measurements that reflect the quality of a user u's profile; then, we try to link the quality of the profile with the quality of the recommendations provided by the recommender to the same user u.

We run our investigation on the ML1M and LT datasets. For each dataset, we use the data split presented in Section 5.3.3. In particular, we analyse user profiles of the training set, K^{before}. For the sake of the exposition here and in the rest of this chapter, we denote with $I_u = \{i \in I : r_{u,i} \in K_u^{before}\}$ the profile of the user u, i.e. the items rated by u in K^{before}; and we denote with $I_u^+ = \{i \in I_u : r_{u,i} \in K_u^{before} \wedge r_{u,i} > 3\}$ the profile of the user u restricted to items that are relevant to u.

For a user $u \in U$, we propose to measure the quality of her profile by using the following metrics:

- size of u's profile, i.e. $|I_u|$;
- diversity of u's profile, in terms of ILD (see Formula 2.4), i.e. $\mathrm{ILD}(I_u)$ for all the items in u's profile and $\mathrm{ILD}(I_u^+)$ for all the relevant items in u's profile;
- novelty of u's profile in terms of PC (see Formula 2.7), i.e. $\mathrm{PC}(I_u)$ for all the items in u's profile and $\mathrm{PC}(I_u^+)$ for all the relevant items in u's profile;
- average rating of u's profile, i.e. $\mathrm{avg}(I_u) = \frac{1}{|I_u|} \sum_{i \in I_u} r_{u,i}$

To measure recommendation quality, we consider the top-10 recommendation list provided to u and we measure the diversity of the list in terms of ILD and EILD (see formulas 2.5 and 2.6); novelty in terms of PC and EPC (see formulas 2.8 and 2.9); and serendipity in terms of CBS and ECBS (see formulas 2.11 and 2.12). We calculate the Pearson coefficient r to measure the correlation between the quality of the user profiles and the quality of the recommendations provided to the users.

Results

Results are reported in Tables 6.1 and 6.2: they mostly reveal a weak or no correlation between the measure of profile quality and the measures of recommender performance. In both datasets, quite strong correlations are found for the ($\mathrm{ILD}(I_u)$, ILD) pair ($r = 0.67$ for ML1M, $r = 0.53$ for LT), meaning that a user profile made of diverse items favours a set of recommendations which is

Table 6.1: ML1M. Pearson coefficients between the quality of the user profiles and the recommender's performance. Each row corresponds to a measure of profile quality, while each column corresponds to a measure of recommender performance. In the table we only report coefficients $r > |0.25|$, i.e. where at least a weak correlation is found.

	Recall	ILD	EILD	PC	EPC	CBS	ECBS
$\|I_u\|$						-0.42	
$\mathrm{ILD}(I_u)$		0.67		-0.26		0.28	
$\mathrm{ILD}(I_u^+)$		0.3					
$\mathrm{PC}(I_u)$				0.65		-0.26	
$\mathrm{PC}(I_u^+)$							
$\mathrm{avg}(K_u^{before})$							

Table 6.2: LT. Pearson coefficients between the quality of the user profiles and the recommender's performance. Each row corresponds to a measure of profile quality, while each column corresponds to a measure of recommender performance. In the table we only report coefficients $r > |0.25|$, i.e. where at least a weak correlation is found.

	Recall	ILD	EILD	PC	EPC	CBS	ECBS
$\|I_u\|$							
$\mathrm{ILD}(I_u)$		0.53				0.42	
$\mathrm{ILD}(I_u^+)$							
$\mathrm{PC}(I_u)$				0.64		0.32	
$\mathrm{PC}(I_u^+)$							
$\mathrm{avg}(K_u^{before})$							

diverse; and for the ($\mathrm{PC}(I_u)$, PC) pair ($r = 0.65$ for ML1M, $r = 0.64$ for LT), meaning that a user profile made of novel (unpopular) items favours a set of recommendations which is novel.

The Tables also show that CBS is weakly influenced by a few measures of profile quality. One of these is the profile size (only for ML1M, $r = -0.42$), i.e. the bigger is a user profile, the less serendipitous are the recommendations made to the user: we argue that, when a user profile is big, it is difficult for the recommender to recommend items which surprise the user (i.e. items that are different from the ones already included in the profile). Moreover, the diversity of a profile influences the serendipity of the recommendations ($r = 0.28$ for ML1M, $r = 0.42$ for LT).

Finally, we note we obtained contradictory results for the novelty of a user profile in relation with the CBS of a recommendation list. In LT, we get a weak positive correlation ($r = 0.32$), i.e. the more unpopular are the items in

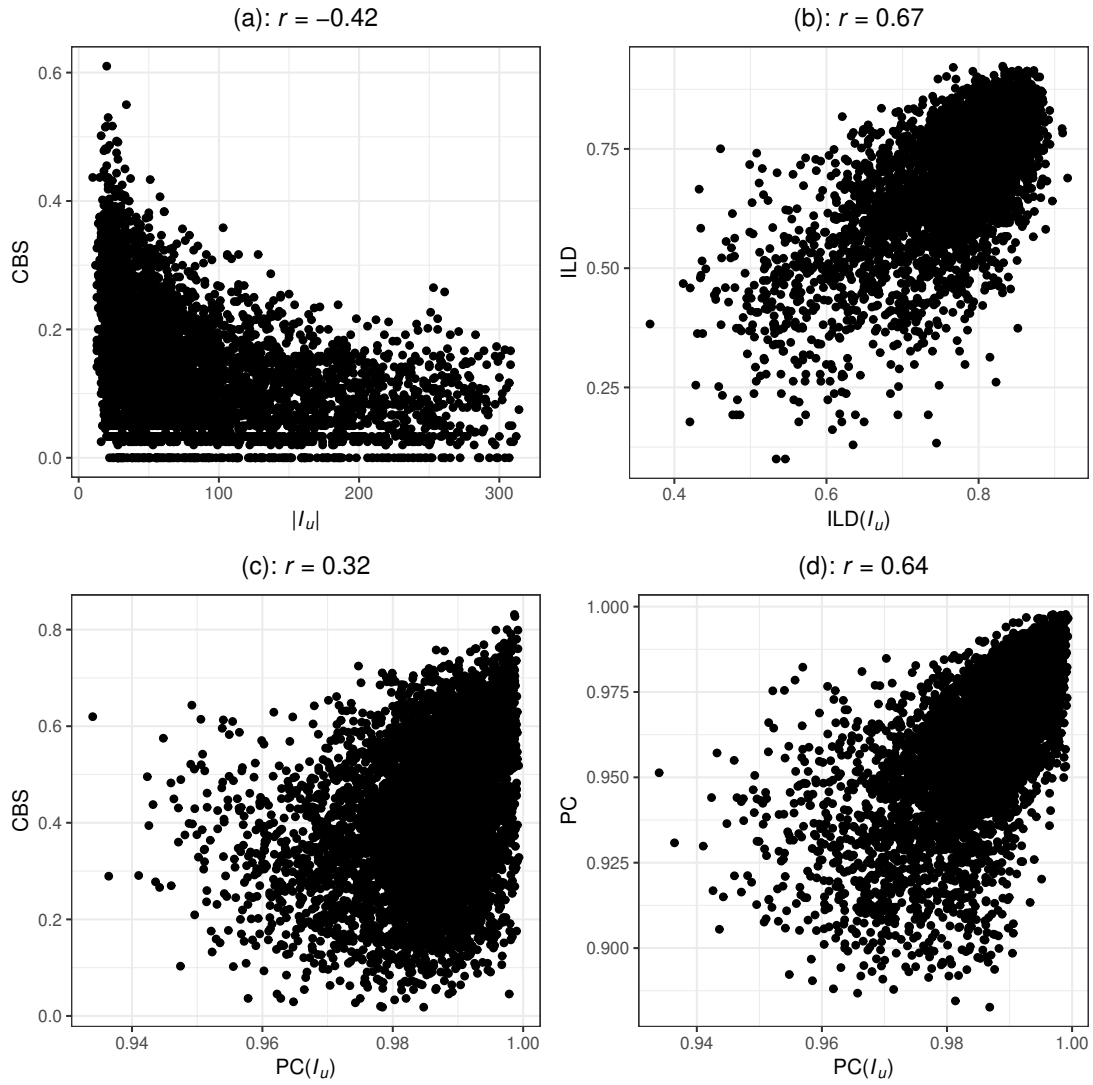

Figure 6.1: For each subplot, on the x-axis is a measure of profile quality; on the y-axis is a measure of recommendation quality. Subplots (a) and (b) are from ML1M; subplots (c) and (d) are from LT.

a user's profile, the more a recommendations list is surprising to the user. But, in ML1M, the opposite is true (the correlation value is similar but negative, $r = -0.26$), i.e. the more unpopular are the items in a user's profile, the less a recommendation list is surprising to the user. We also note that none of the measures of profile quality seem to influence the Recall. Furthermore, the novelty of profiles calculated on only relevant items and the average ratings of the profiles do not seem to have any bearing on any recommendation quality.

In Figure 6.1, we show scatter plots that correspond to four of the entries from Tables 6.1 and 6.2, choosing entries where the quality of the profiles is somewhat correlated with recommendation quality.

Conclusions

The motivation behind this investigation was to get some insight into how to design a new effective AL strategy. In particular, we explored the idea that improving a user profile can improve the recommendations provided to the user. As a consequence, an AL strategy that improves a user profile indirectly improves the recommendations provided to the user. We proposed six different ways of modelling the quality of a user profile, and we analyse their influence on the quality of the recommendations.

The results of the investigation revealed that the diversity and the novelty of the user profile are the qualities that most impact the recommendations made to the user. In particular, although accuracy (in the form of Recall) seems not to be affected by any measure of profile quality, we have that the diversity, novelty and serendipity of recommendations are indeed impacted to some extent. In the next sections, we will use these findings when designing new AL strategies.

6.1.2 Evaluating a strategy under an artificial setting

In Section 5.1.2 we described an AL evaluation methodology under what we called an artificial setting as an alternative to what we called an evaluation under a realistic setting (the latter being what we employed in the experiments of Section 5.3). Under the artificial setting, the methodology sets the candidate items set of a user u to be her set of hidden items, i.e. $C_u = H_u$. Practically, this means that a user u will always be able to provide the rating for an item when asked, as if the strategy knew beforehand which ratings are known by the user. A consequence is that, for queries where $|Q_u| = n_q$, for a given user, different strategies will elicit the same number of ratings.[1]

Although unrealistic, the artificial setting is useful for at least two reasons. First, it is complementary to evaluation under the realistic setting. Evaluation under the realistic setting reveals how good an AL strategy is at eliciting ratings in terms of *quantity* (how many ratings) and *quality* (how useful they are at improving the recommender's performance). Under this setting, it is easy to establish how many ratings are elicited by each strategy (and therefore, which strategy is the best at doing that). However, more difficult is to determine the quality of the elicited ratings, because performance will be affected by quantity

[1] A particular user u gives up to n_q ratings (indeed, u might have fewer than n_q ratings in H). However, the experiment is fair to all strategies compared because they elicit the same number of ratings for the same user.

as well (see the discussion in Section 5.4). On the other hand, in the artificial setting each strategy elicits the same number of ratings, and so the evaluation allows one to answer the following question: "*what would the performance changes brought by different AL strategies be like, if they all had the possibility to elicit the same number of ratings?*".

The second main advantage of the artificial setting is that it speeds up the running of the experiments themselves because the candidate items sets are very much smaller than the ones in the realistic setting. Indeed, recalling the Query Selection Algorithm 1, an AL strategy is now required to score $|H_u|$ items in each loop iteration (see line 3) when composing the query for the user u, i.e. $|C_u| = |H_u| << |I|$.

Evaluation under the artificial setting can be used in a preliminary stage of the entire offline evaluation. In this chapter, we will make use of it to reveal the quality of an AL strategy when having lots of new strategies to compare; we will filter out those that do not show good performance improvements. When using the artificial setting, each AL strategy will produce queries for five items, i.e. $n_q = 5$. When tuning an AL strategy's hyperparameters (Section 5.2.4), we will be consistent and use the artificial setting for that stage too.

The ideal AL strategy

When evaluating AL strategies, we might be interested in the highest performance achievable, i.e. an upper bound. We can refer to this as the performance obtained by a fictitious 'ideal' AL strategy. It might be though that this 'ideal' AL strategy will be one that is able to elicit all the ratings in the hidden set. But, this is not the case. A recommender that is built on a training set that additionally includes all the hidden ratings might actually perform worse than one built on a training set that includes a subset, including none, of the hidden ratings. For example, this might happen when some of the elicited ratings worsen the recommender's performance, instead of improving it: perhaps they are noisy, for example. In other words, we argue that the 'ideal' AL strategy is not necessarily the one able to elicit all the ratings in the hidden set, rather it is the one that improves the recommender by the greatest extent, regardless of the number of ratings elicited. Most of the time the performance upper bound is unknown because the ideal strategy is unknown. A procedure to reveal this upper bound is to train recommenders on a training set augmented by every possible subset of the hidden ratings. Unfortunately, this procedure is unfeasible in practice.

Table 6.3: Percentage differences between the omniscient strategy's performance in comparison with HP (under the *INT_HT* evaluation methodology)

	ML1M		LT	
	omniscient	HP	omniscient	HP
Recall	15.27%	10.39%	16.35%	8.91%
EILD	36.93%	23.59%	46.75%	20.24%
EPC	17.99%	13.20%	20.20%	10.70%
ECBS	7.35%	6.60%	20.26%	7.30%
Avg. # of elicited ratings	14.13	2.10	10.44	0.65

.

However, the performance of a recommender built on a training set that includes all the hidden ratings does establish a reference point for the evaluation of new AL strategies. Let's refer to an AL strategy that can elicit all the hidden ratings as omniscient.[2] In Table 6.3, we show the performance achieved by this omniscient strategy for Active users. (We omit results on the system-wide perspective, as they are similar.)

As expected, performances achieved by the omniscient strategy are better than the ones obtained by HP, even though it was the best strategy in the experiments in the case study of Section 5.3. However, for ML1M, despite the huge difference in the number of ratings elicited (about 2 ratings per Active user for HP and about 14 for omniscient), their performances are quite close to each other. For LT, the difference in the number of elicited ratings is even bigger between the two strategies and so is the difference in their performance. As stated before, we will use performances obtained by the omniscient strategy as a reference, keeping in mind also that, after all, HP is still a good strategy.

6.2 Strategies Targeting Diversity

In this section, our goal is to design new strategies that will help the recommender to provide recommendation lists that are more diverse. In Section 6.1.1, we found that the diversity of a user profile impacts the diversity of the recommendations: thus, increasing the diversity of the users' profiles is the focus of the new strategies that we propose.

- Diversity-For-Profile (Div4P): The goal of the strategy is to elicit ratings

[2] As per the discussion of the preceding paragraph, the strategy is omniscient only in the sense that it knows which ratings it will obtain; it does not know what subset of these ratings will most improve recommender performance.

for items that will increase the diversity of a user profile I_u. The strategy uses $f_S(u, i, G = I_u) = \text{ILD}(I_u \cup i)$ to score candidate items. To note, this strategy can speed up the computation of Q_u by scoring each candidate item just once, because f_S does not depend on Q_u.

- Diversity-For-Profile-Relevant (Div4PR): This is similar to Div4P, but this time a user profile is composed of relevant items only, i.e. I_u^+. Thus, the strategy uses $f_S(u, i, G = I_u^+) = \text{ILD}(I_u^+ \cup i)$ to score candidate items. (Again, scoring each candidate item can be done just once.)

- Diversity-For-Profile-Diversity-For-Query (Div4P-Div4Q): One goal of this strategy is the same as Div4P, i.e. to increase the diversity of a user profile I_u. But, additionally, another goal is to diversify also the query Q_u itself. We argue that similarly to a diverse recommendation list, a diverse query might increase the chance of obtaining ratings. To the best of our knowledge, only the approach of Kohrs and Merialdo in the RS literature tries to diversify the query items (and they use a ratings-based similarity metric as a proxy for items diversity) [KM01]. Thus, we design a strategy which combines two scores for a candidate item i: the score of Div4P and a second score which measures the diversity of the query if i is included; the final score for i is given by a linear combination of the two scores, controlled by a parameter λ. $f_S(u, i, G = I_u, Q_u) = \lambda \,\text{ILD}(I_u \cup i) + (1 - \lambda)\,\text{ILD}(Q_u \cup i)$.

- Diversity-For-Profile-Relevant-Diversity-For-Query (Div4PR-Div4Q): This is similar to Div4P-Div4Q, but this time a user profile is composed of relevant items only. Thus, the strategy uses $f_S(u, i, G = I_u^+, Q_u) = \lambda \,\text{ILD}(I_u^+ \cup i) + (1 - \lambda)\,\text{ILD}(Q_u \cup i)$ to score candidate items.

All four strategies are personalised. Div4P-Div4Q and Div4PR-Div4Q have a hyperparameter λ which values are selected from $V_\lambda = \{0, 0.1, 0.3, 0.5, 0.7, 0.9\}$: we tune it following the same procedure described in Section 5.2.4 and we optimise again for Recall. Although it might have made sense to choose to optimize a diversity metric instead of Recall (because such strategies are supposed to target diversity), we argue that increasing accuracy remains the main objective of an AL strategy.

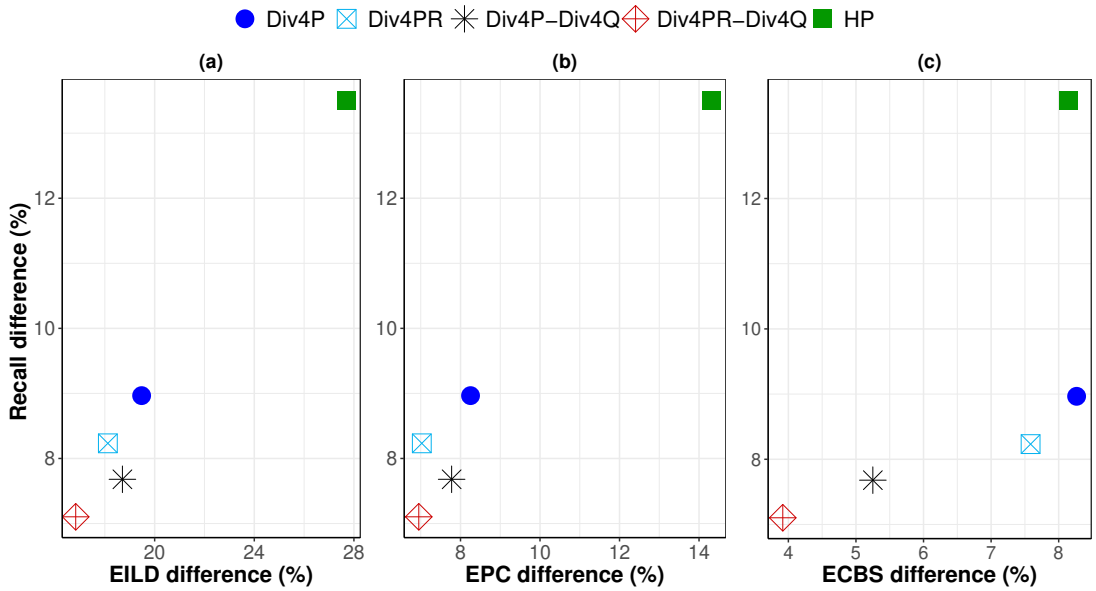

Figure 6.2: ML1M results for strategies targeting diversity (Active users, artificial setting).

6.2.1 Results

6.2.1.1 Experiments under the artificial setting

Following the methodology described in Section 6.1.2, we first run experiments under the artificial setting, to reveal the potential benefit of the four designed strategies (and we use the same data splits we obtained in 5.3.3). Figures 6.2 and 6.3 show results for Active users for both ML1M and LT. (Note that, in the artificial setting, Active users and Respondents are now the same set of users because all users provide the same amount of ratings under this setting). HP consistently outperforms all four new strategies, except for one case in ML1M, where Div4P is competitive with HP according to ECBS.

6.2.1.2 Experiments under the realistic setting

Results for experiments under the realistic setting are not shown here because all four strategies are inferior to HP. We argue this is mainly due to the low number of ratings acquired by all four new strategies; we obtain ratings from only a few Respondent users (i.e. around 50 users on average).

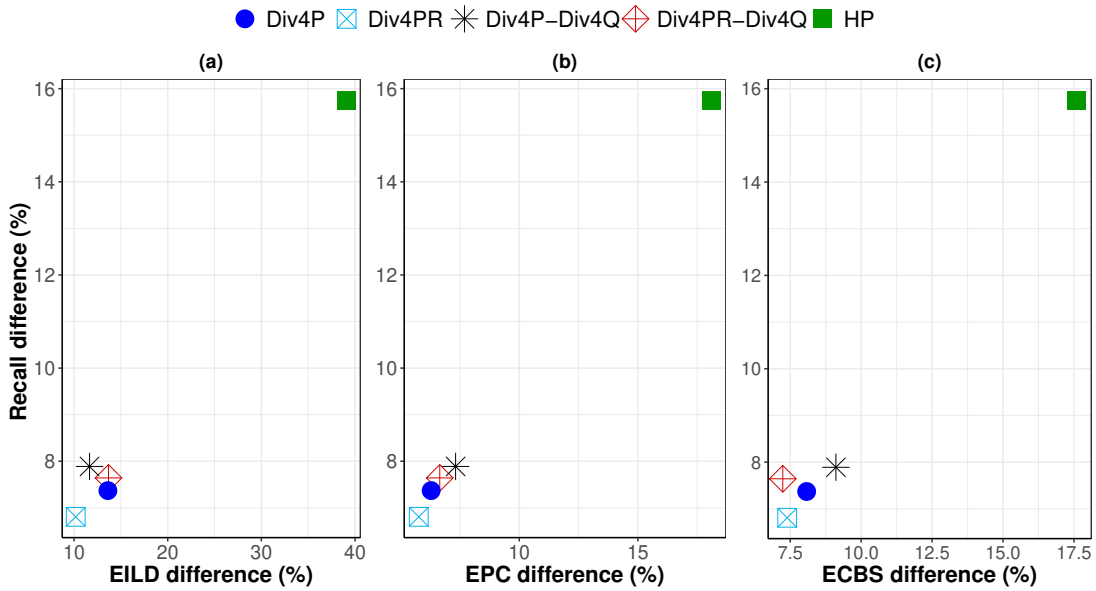

Figure 6.3: LT results for strategies targeting diversity (Active users, artificial setting).

6.2.2 Conclusions

In this section, we explored the idea of diversifying the query of a user (i.e. so that the query is composed of a set of diverse items) in addition to asking the users to rate items which make their profile more diverse. However, both ideas fail to rival HP both in improving the diversity of recommendations and also in improving the other accuracy and beyond-accuracy measures. However, in Section 6.5, we will propose a hybrid strategy that leverages these ideas to design new strategies.

6.3 Strategies Targeting Novelty

In Section 2.1 we defined novelty in terms of popularity by using the Popular Complement (PC) metric: the less popular among users a set of items is, the greater is the score of this metric (see Formula 2.7). Section 6.1.1 suggests that the novelty of a user profile is positively correlated with the novelty of the recommendations received by the same user; and that this is also true for the serendipity of the recommendations for LT (although there is a negative correlation with serendipity of the recommendations for ML1M).

Therefore, an effective strategy aiming at increasing novelty might be one that explores items and tastes that are less popular among the users in the sys-

tem. It will be interesting also to see the effect of this strategy on the other beyond-accuracy metrics, especially for serendipity. We propose three different strategies:

- Item-Novelty (ItemNov): The goal of this strategy is to elicit ratings for items that are unpopular (i.e. less rated), in an attempt to explore whether a user is aware of the parts of the item catalogue ignored so far by most of the users. The strategy uses $f_S(u, i, G = I_u) = 1/|U_i|$ to score candidate items, with $U_i = \{u \in U : r_{u,i} \in K^{before}\}$ being the set of users that have rated the item i in K^{before}. Note that this strategy can speed up the computation of Q_u by scoring each candidate item just once, because f_S does not depend on Q_u.
- Features-Novelty-avg (FeatNov-avg): The goal of this strategy is to elicit ratings for items that have unpopular item features, by which we mean those features that belong to items that have fewer ratings. We denote with $f \in \mathcal{F}$ a generic item feature in the feature space $\mathcal{F}$ and with $\mathcal{F}_i \subseteq \mathcal{F}$ the set of features of the item i. The strategy uses $f_S(u, i, G = \mathcal{F}) = (\underset{f \in \mathcal{F}_i}{avg}|\{j \in I : f \in \mathcal{F}_j\}|)^{-1}$ to score candidate items, where we use the inverse to penalise items with popular features. Note that, because f_S does not depend on Q_u, this strategy can speed up the realisation of Q_u by scoring each candidate item just once.
- Features-Novelty-max (FeatNov-max): The goal of this strategy is the same as that of FeatNov-avg above. The only difference is that FeatNov-max considers the inverse of the maximum popularity instead of the average in FeatNov-avg, i.e. $f_S(u, i, G = \mathcal{F}) = (\underset{f \in \mathcal{F}_i}{max}|\{j \in I : f \in \mathcal{F}_j\}|)^{-1}$. Once again, this strategy can speed up the computation of Q_u by scoring each candidate item just once because f_S does not depend on Q_u.

None of these strategies is personalized (although the queries might be different just because the candidate set is slightly different) and none of them has hyperparameters.

6.3.1 Results

6.3.1.1 Experiments under the artificial setting

Figures 6.4 and 6.5 show results of the evaluation under the artificial setting for Active users. HP consistently outperforms all three new strategies, except

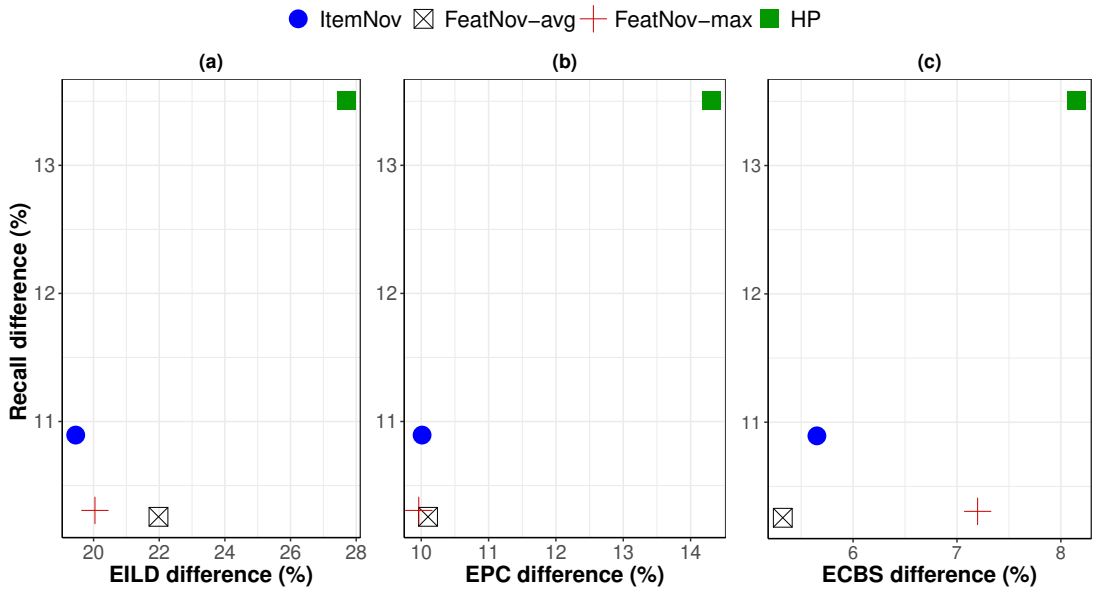

Figure 6.4: ML1M results for strategies targeting novelty (Active users, artificial setting).

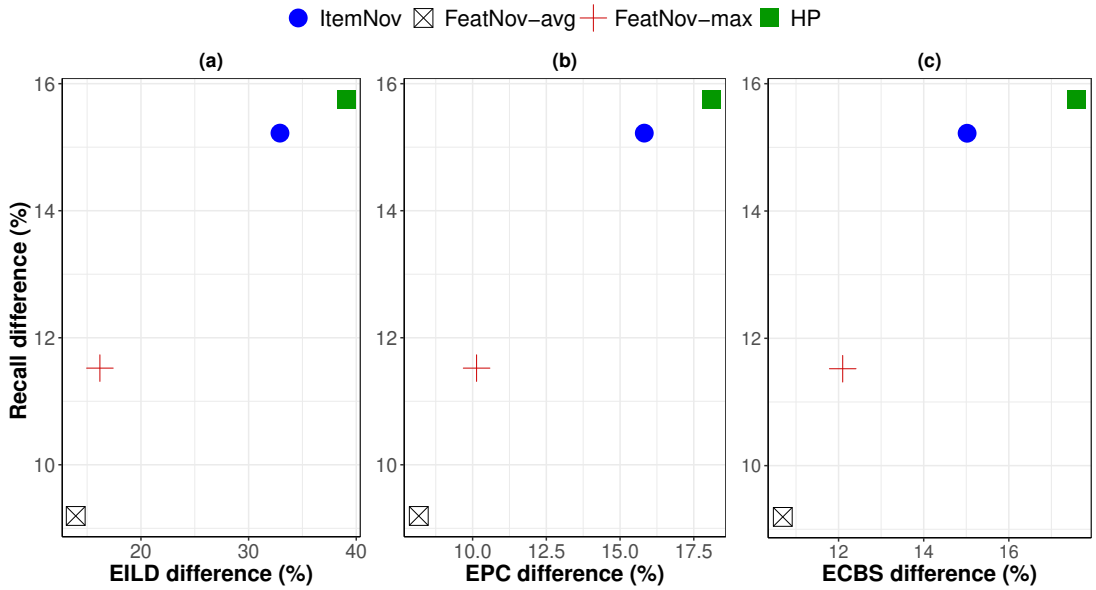

Figure 6.5: LT results for strategies targeting novelty (Active users, artificial setting).

for one case in LT, where ItemNov is competitive with HP accordingly to Recall.

6.3.1.2 Experiments under the realistic setting

Results for experiments under the realistic setting methodology are not shown here: HP outperforms all three new strategies and we argue that this happens for the same reason that we gave in Section 6.2.1.2.

6.3.2 Conclusions

In this section, we explored the idea of eliciting items which are unpopular or that have unpopular features to increase the novelty of the recommendations. However, HP still outperforms all the new strategies for all the performance metrics. In Section 6.5, we will propose a hybrid strategy that uses these ideas to design new strategies.

6.4 Strategies Targeting Serendipity

In Section 2.1 we defined serendipity in terms of Content-Based Surprise (CBS): a serendipitous item for a user is one that is unlike any item the user has seen before, i.e. one that is different to the items in her profile. This time we have no insights from Section 6.1.1. Our idea for an AL strategy targeting CBS is one that attempts to discover new tastes for a user, selecting items which are different from the ones already in the user profile. We therefore design two new strategies:

- Surprise-To-Profile-min (Surp2P-min): The goal of this strategy is to elicit ratings for items that are different to a user's profile in an attempt to explore new and possibly surprising items that will enrich her profile. For each candidate item, this strategy calculates how surprising the candidate item is with respect to the items in u's profile, i.e. $f_S(u, i, G = I_u) = \min_{j \in I_u} \text{dist}(i, j)$. Because f_S does not depend on Q_u, this strategy can speed up the computation of Q_u by scoring each candidate item just once.
- Surprise-To-Profile-avg (Surp2P-avg): This strategy is similar to Surp2P-min, but instead of the lower bound, it uses the average of the item's distance from the user profile items as an indicator of surprise (in line with [VC11, AT14a], for example), i.e. $f_S(u, i, G = I_u) = \underset{j \in I_u}{avg}\, \text{dist}(i, j)$. Because f_S does not depend on Q_u, this strategy can speed up the realisation of Q_u by scoring each candidate item just once.

Note that the distance function $\text{dist}(i, j)$ that we use is the Jaccard distance between items i and j calculated on the item's features; and both strategies are personalized.

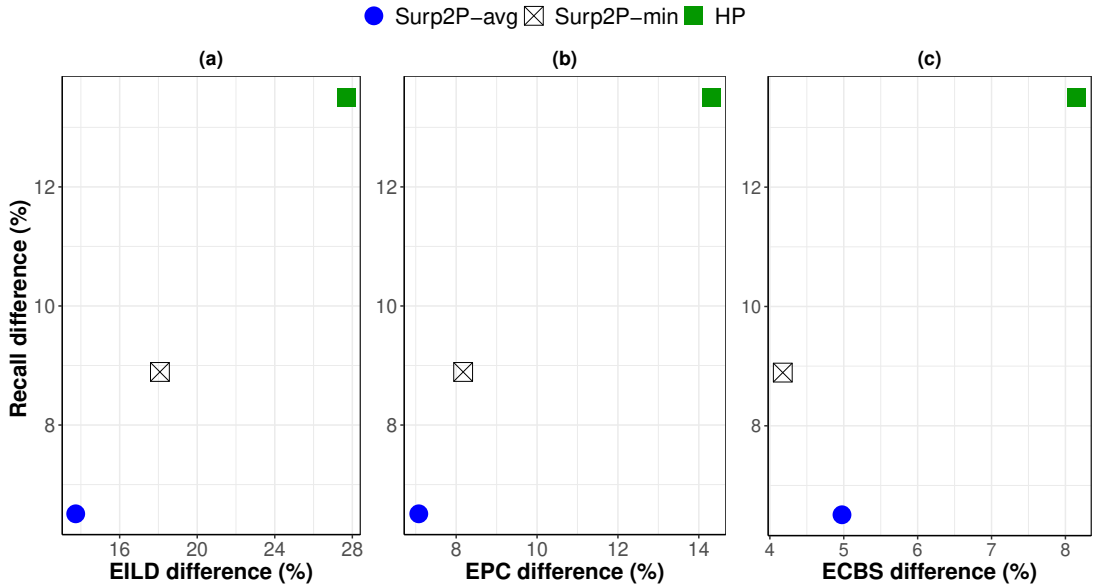

Figure 6.6: ML1M results for strategies targeting serendipity (Active users, artificial setting).

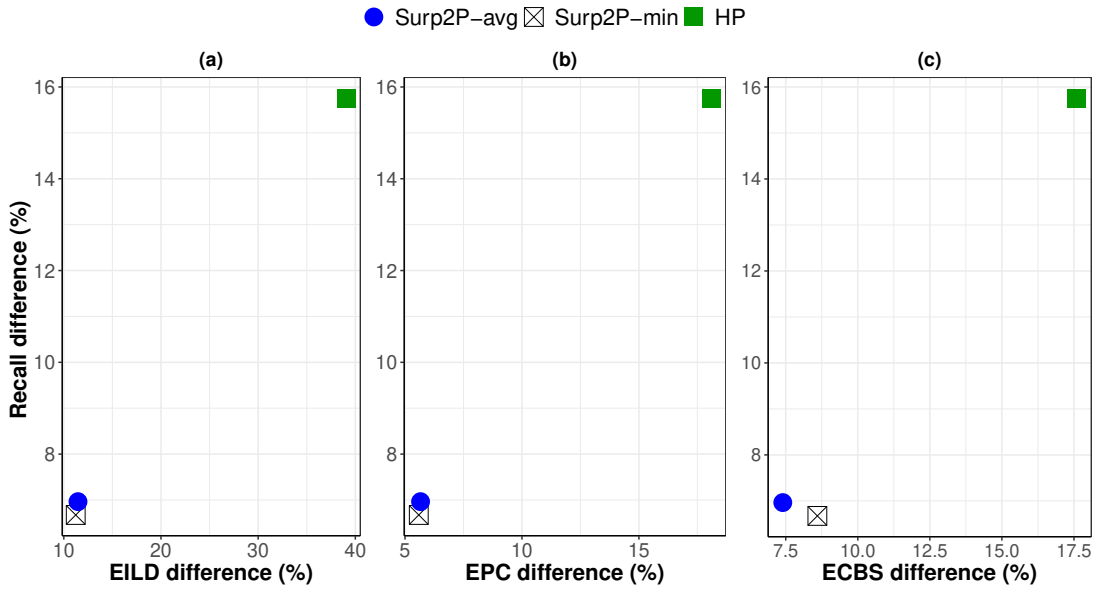

Figure 6.7: LT results for strategies targeting serendipity (Active users, artificial setting).

6.4.1 Results under the artificial setting

Figures 6.6 and 6.7 show results under the artificial setting for Active users. HP consistently outperforms both new strategies in both datasets according to all metrics.

6.4.2 Experiments under the realistic setting

Results for experiments under the realistic setting are not shown here. Once again, HP outperforms both new strategies and we argue that this happens for the same reason that we gave in Section 6.2.1.2.

6.4.3 Conclusions

In this section, we explored the idea of eliciting items which are serendipitous to a user, in an attempt to enrich a user profile with new items, different from the ones already rated so far. However, HP still outperforms all new strategies for all metrics. In Section 6.5, we will propose a hybrid strategy that uses these ideas to design new strategies.

In the next section we will combine ideas from Sections 6.2, 6.3 and 6.4 to build new strategies that are more successful at improving the recommender's performance.

6.5 Hybrid Strategies

In this section, we explore a hybridization approach to AL strategies, introduced earlier in Section 4.4: a strategy of this kind combines criteria from different strategies to build the query. Hybridization can be obtained by using different schemes, described in works such as [Bur02] and [ERR16]. In [Bur02], Burke surveys hybrid recommenders; in [ERR16], Elahi et al. survey hybrid methods applied to Active Learning in recommender systems. Inspired by these works, we designed and implemented two hybridization schemes. In the following, we describe these schemes, where each AL hybrid strategy S combines two individual strategies, $S1$ and $S2$, with f_{S1} and f_{S2} being their scoring functions respectively.

- *Weighted*: f_{S1} and f_{S2} compute scores for the candidate items set C_u individually. Then, those scores are combined together to produce a single final score. For each $i \in C_u$, the final score is obtained by applying a linear combination of the two independent scores from f_{S1} and f_{S2}, i.e. $f_S(i) = \lambda f_{S1}(i) + (1 - \lambda) f_{S2}(i)$. The hyperparameter λ controls the weight of the single strategies on the final score, so that scores reflect the importance of every single strategy in the selection. The main drawback of this hybridization scheme is that, when n strategies are combined, n different

hyperparameters are needed.

- *Cascade*: The query for a user u is produced by means of a two-stage process where, informally, one strategy refines the query given by the other strategy. More formally, in the first stage the strategy $S1$ produces the query Q_u^{S1} of size $|Q_u^{S1}| = M > n_q$, following the usual algorithm (Algorithm 1). In the second stage, the candidate items set C_u^{S2} that is given to $S2$ contains only the items in Q_u^{S1}, and $S2$ produces the final query Q_u using Algorithm 1. In other words, $S1$ is the baseline strategy which decides which items are promising to be included in the final query; $S2$ then decides how to compose the final query selecting from those items. This scheme has a hyperparameter to tune, i.e. M, which somehow controls the weight of each strategy: when M is small, i.e. $M \rightarrow n_q$, the final query strongly depends on $S1$'s selection criteria; when M is large, i.e. $M \rightarrow |C_u|$, the final query strongly depends on $S2$'s selection criteria. A cascade allows us to filter the whole candidate set of items using one criterion, while using a second criterion to refine the query on the remaining items. It turns out that deciding which strategy plays the role of $S1$ and which one plays the role of $S2$ heavily affects the results of the hybridization under this scheme.

For notation, when we do not need to distinguish between the weighted and the cascade, we denote the hybrid by $S1\&S2$. When differentiation is needed, we will denote the weighted scheme by $S1 + S2$ and the cascade by $S1 > S2$. When creating a weighted hybrid, it does not matter which of two AL strategies plays the part of $S1$ and which is $S2$. In a cascade, on the other hand, the choice is significant.

Each new strategy that we present in the following is the combination of HP and one of the strategies presented in previous sections of this chapter. The choice of HP is straightforward: in our experiments, it has been the best strategy for both datasets, effective at eliciting a large number of ratings and improving the recommender's performance (see also the discussion in Section 6.1.2). Additionally, HP is computationally fast at calculating scores for the candidate items, since it uses scores given by the recommender itself.

HP is paired with a second strategy and its choice is justified based on the results obtained in the previous sections on the two datasets. In general, we pick the strategies that are successful at improving both Recall and the metric which they were designed for. Among the strategies targeting diversity, we

employ Div4P and HP&Div4P-Div4Q; among the ones targeting novelty, we employ HP&ItemNov; and we use both HP&Surp2P-avg and HP&Surp2P-min for serendipity. In cascades, HP will take the role of $S1$ and the other strategy takes the role of $S2$. We choose HP to be $S1$ so that HP can effectively identify the items which are known and useful in first place. Then, $S2$ can refine the query by choosing the most useful items according to its own criteria.

In the following list, we present our new strategies:

- HP&Div4P, i.e. the hybridization of HP and Div4P.
- HP&Div4P-Div4Q, i.e. the hybridization of HP and Div4P-Div4Q.
- HP&Div4Q i.e. the hybridization of HP and Div4Q. Similarly to Section 6.2, Div4Q measures the diversity of the query if i is included, using $f_{Div4Q}(i) = \text{ILD}(Q_u \cup i)$.
- HP&ItemNov, i.e. the hybridization of HP and ItemNov.
- HP&Surp2P-avg, i.e. the hybridization of HP and Surp2P-avg.
- HP&Surp2P-min, i.e. the hybridization of HP and Surp2P-min.

Hyperparameter values for λ (of the weighted hybridization) and for M (of the cascade hybridization) of every different strategy are selected from $V_\lambda = \{0, 0.1, 0.3, 0.5, 0.7, 0.9\}$ and $V_M = \{100, 300, 500\}$ respectively, again following the procedure described in Section 5.2.4 where we optimise for Recall.

To evaluate the effectiveness of these new strategies, we evaluate under the realistic setting using the data splits obtained in 5.3.3. For comparison, we include HP's results, which we obtained in Section 5.3. In the following, we report and analyse the most significant results of these experiments.

Results for ML1M

In this section, we analyse results obtained on ML1M by the new hybrid strategies. As Table 6.4 shows, HP is still the strategy that elicits the largest number of ratings and by more users than the other strategies do. The closest strategy is the hybrid HP&Div4Q, in both weighted and cascade versions (with weighted slightly better than cascade). The rest of the strategies elicit less than half the ratings that HP does, with HP&Div4P-Div4Q being by far the worst.

Figures 6.8 and 6.9 show the performances of weighted and cascade hybrid strategies respectively for Active users and for the System-wide perspective.

Table 6.4: ML1M. The table reports, for each strategy and for each hybridization approach, the average number of elicited ratings, the average number of elicited ratings per Active User, and the average number of users who are Respondents. Averages are calculated across the 10 folds.

	# elicited	# elicited per user	# Resp.	# elicited	# elicited per user	# Resp.
HP	3107.6	2.12	1241.7	3107.6	2.12	1241.7
	Weighted			Cascade		
HP&Div4P	1485.2	1.01	691.6	1115.7	0.76	655.1
HP&Div4Q	3060.9	2.09	1240.7	1789.7	1.22	969.4
HP&Div4P-Div4Q	172.9	0.12	153.2	98.4	0.07	92.3
HP&ItemNov	1385.1	0.94	578	887	0.60	638.6
HP&Surp2P-min	881.0	0.60	470.7	1377.8	0.94	778.2
HP&Surp2P-avg	1113.8	0.76	561.5	596.4	0.41	366.9

The heat maps confirm the statistics of Table 6.4, where once again the strategies that elicit more ratings are the ones that improve the recommender the most. For Active users, the few strategies that achieve a statistically significantly different performance after the AL iteration (see Tables A.10 and A.12 in the Appendix) are: HP and HP&Div4Q (weighted and cascade) which bring the best improvement to the recommender, according to all metrics; HP&Div4P (weighted and cascade) and HP+ItemNov which obtain very similar performances but way behind the ones of HP and HP&Div4Q. Among those, HP can still be considered the best strategy because its performance is the best for all metrics and it is statistically significantly different from the performance of all the other strategies. Similar outcomes stand for the System-wide perspective (see Tables A.11 and A.13 in the Appendix for the statistical significance tests).

We continue our analysis considering only a subset of all the strategies presented so far, i.e. the best-performing ones. Therefore, we keep HP and the two versions of HP&Div4Q, along with HP+Div4P and HP+ItemNov.

Figure 6.10 shows results for this group of strategies for Respondents only. For such users, HP is the strategy that elicits the greatest number of ratings (on average) and achieves the best improvements across all metrics, except for EILD, where HP+ItemNov slightly outperforms HP. Regarding the other strategies, the Figure suggests that HP+Div4Q is the second-best strategy for Respondents, while HP+ItemNov is also a competitive strategy.

Like in Section 5.3, we analyse the impact of the best-performing hybrid strategies on the Active users grouped by profile size (grouping them in the same way as before). We report the statistical significance tests of buckets in Table A.18

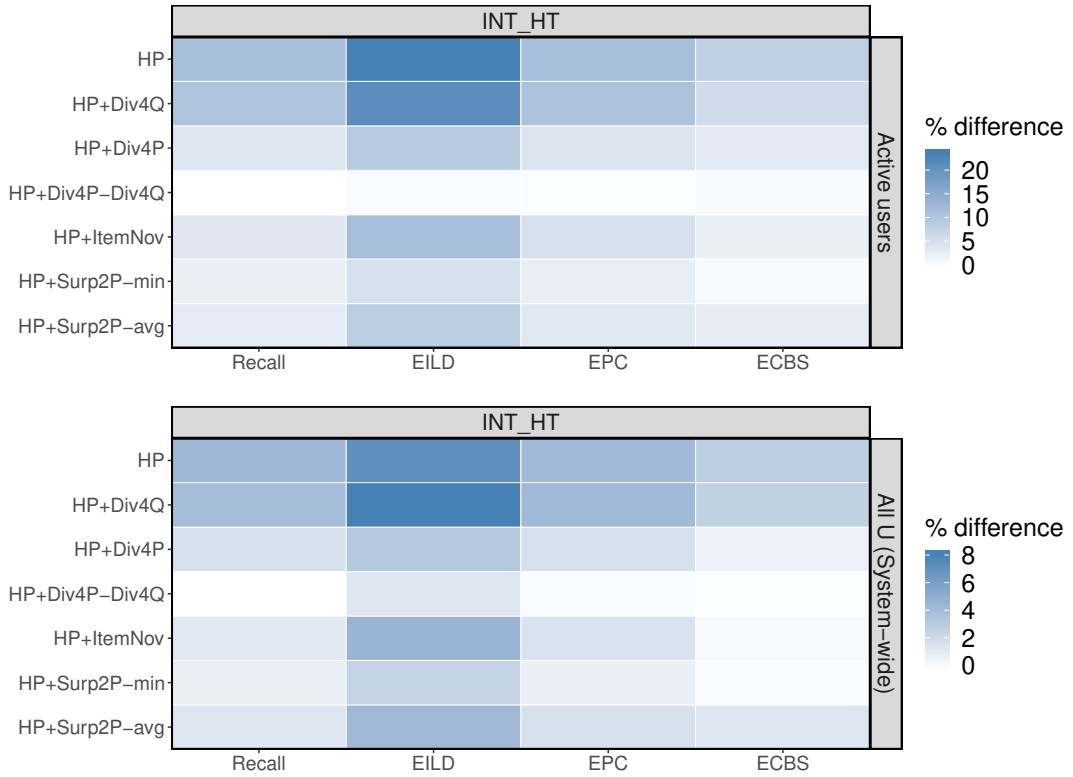

Figure 6.8: ML1M. The heat maps for ML1M show results of the weighted hybrid strategies on two groups of users, i.e. Active Users and U.

in the Appendix. First, we find out that, when grouping users into buckets, results for HP+ItemNov are not statistically different after the AL iteration: thus, we remove this strategy for the rest of the analysis. Also, the last three buckets, i.e. 151-200, 201-300, >300 don't show statistically significant results. In the following, we will give an overview of the most important findings of the remaining results.

For cold-start users, again HP is the best strategy according to all metrics except for EILD, where the best strategy is HP+Div4Q. However, the performance of HP, HP+Div4Q and HP+Div4P are not statistically different from each other for all metrics. We can conclude that all three benefit cold-start users to the same extent.

For mature users, HP seems the best strategy for the first bucket (i.e. 21-50), but again performances are not statistically significantly different from HP+Div4Q and HP+Div4P. In the rest of the buckets, no strategy is consistently the best one, and this is confirmed by the statistical significance tests.

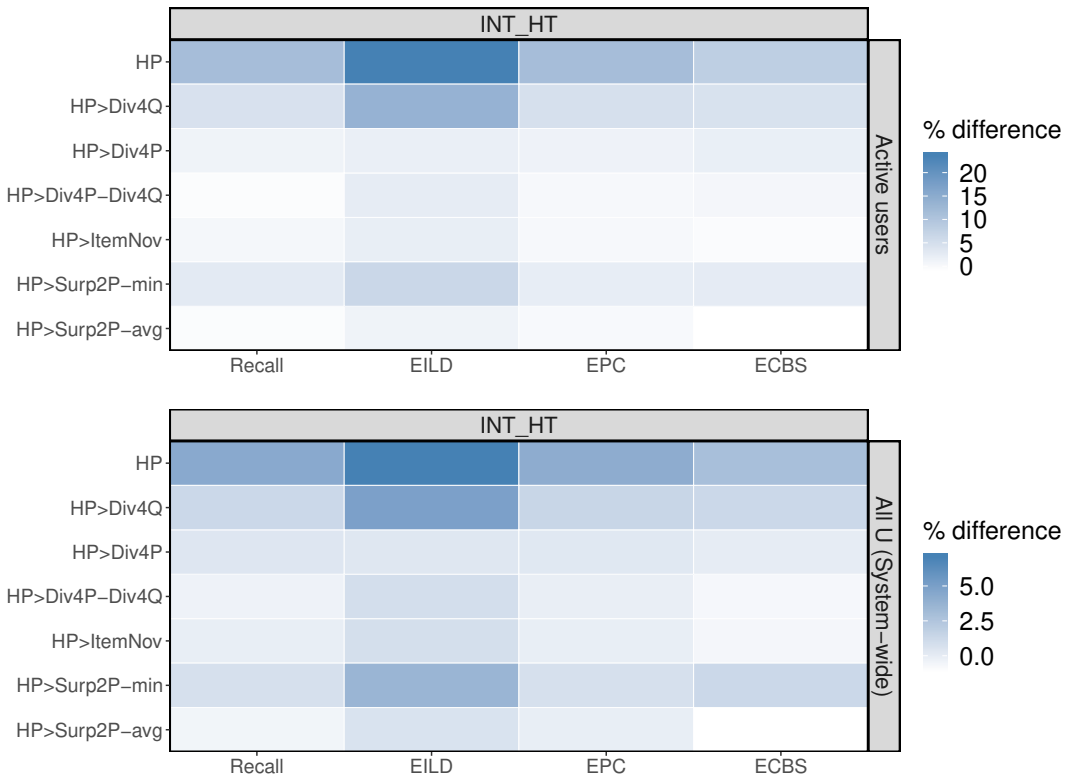

Figure 6.9: ML1M. The heat maps for ML1M show results of the cascade hybrid strategies on two groups of users, i.e. Active Users and U.

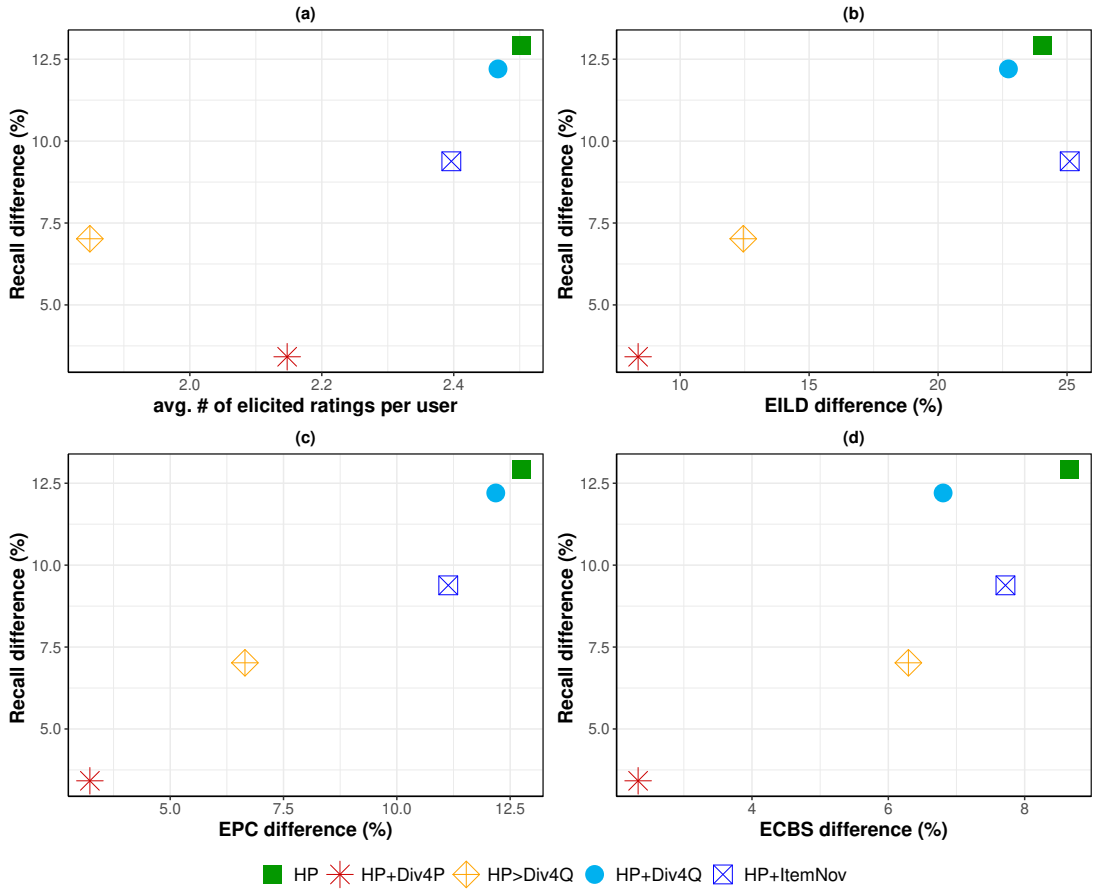

Figure 6.10: ML1M. Respondent users results for the best-performing hybrid strategies.

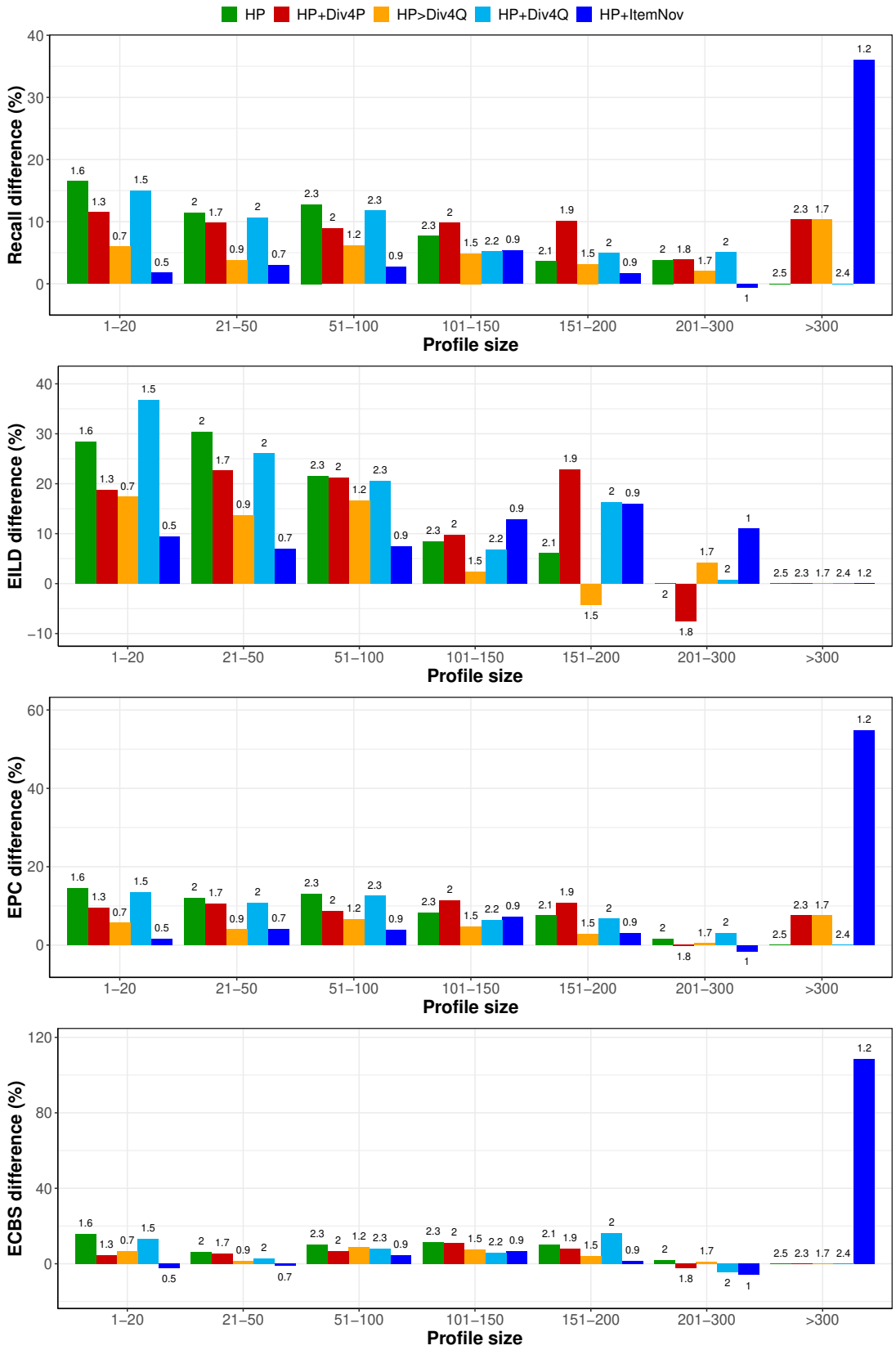

Figure 6.11: ML1M. Active users results by profile size, best-performing hybrid strategies.

Table 6.5: LT. The table reports, for each strategy and for each hybridization approach, the average number of elicited ratings, the average number of elicited ratings per Active User and the average number of users who are Respondents. Averages are calculated across the 10 folds.

	# elicited	# elicited per user	# Resp.	# elicited	# elicited per user	# Resp.
HP	440.4	0.65	679.4	1156.4	0.65	679.4
	Weighted			Cascade		
HP&Div4P	24.1	0.01	23.2	380.6	0.21	290.5
HP&Div4Q	1141.8	0.64	678.1	672.9	0.38	485.7
HP&Div4P-Div4Q	105.3	0.06	96.8	0.0	0.00	0
HP&ItemNov	420.3	0.24	266.3	72.5	0.04	58.2
HP&Surp2P-min	12.9	0.01	12.7	134.3	0.08	111.8
HP&Surp2P-avg	5.4	0.003	4.6	147.6	0.08	126.4

Results for LT

Table 6.5 reports the results obtained on LT: similarly to ML1M, the strategies that elicit a substantial number of ratings are HP, HP&Div4Q (weighted and cascade), HP+ItemNov and, differently to ML1M, HP>Div4P.

Figures 6.12 and 6.13 show performances of weighted and cascade hybrid strategies respectively, for Active users and for the System-wide perspective. The heat maps confirm the statistics of Table 6.5, where once again the strategies that elicit more ratings are the ones that improve the recommender the most. For Active users, only HP, HP&Div4Q (weighted and cascade), HP+ItemNov and HP>Div4P achieve a statistically significantly different performance after the AL iteration (see Tables A.14 and A.16 in the Appendix). Among these, HP and HP+Div4Q clearly show superior performances (and indeed their performances are not statistically significantly different from each other). Similar outcomes stand for the System-wide perspective (see Tables A.15 and A.17 in the Appendix for the statistical significance tests).

For LT, differently from ML1M, the subset of best performing strategies is composed of HP, HP&Div4Q (weighted and cascade), HP+ItemNov and HP>Div4P. Figure 6.14 shows results of this group of strategies for Respondents only. For such users, HP is the strategy that achieves the best improvements across all metrics, and the second-best strategy is HP+Div4Q (which is close to HP).

We again analyse the impact of the best-performing hybrid strategies on the Active users grouped by profile size, similarly to what we have done for ML1M (and we report the statistical significance tests in Table A.19). Also here, we remove HP+ItemNov for the rest of the analysis because its results are not sta-

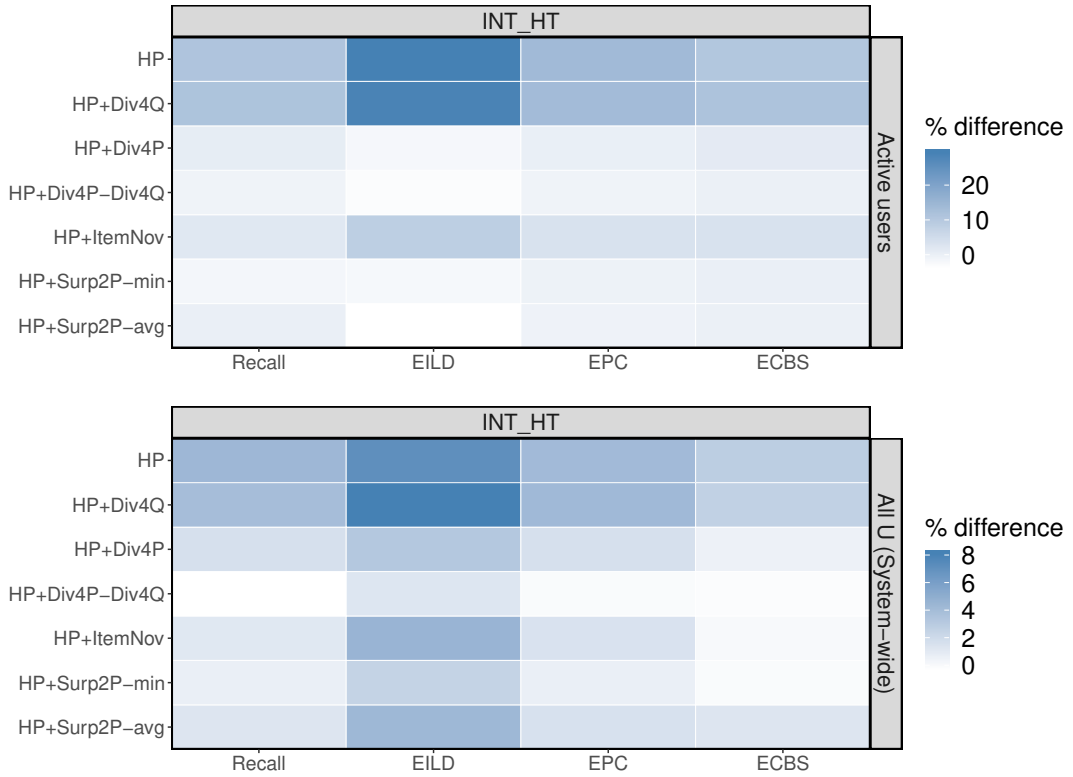

Figure 6.12: LT. The heat maps for LT show results of the weighted hybrid strategies on two groups of users, i.e. Active Users and U.

tistically different after the AL iteration. For cold-start users, HP+Div4Q is the best strategy accordingly to all the metrics. However, results for these buckets are not statistically significant. For mature users, results for the last four buckets (i.e. 101-150, 151-200, 201-300, >300) are almost entirely non-significant; thus, we remove those buckets from the analysis. For the buckets 21-50 and 51-100, HP and HP+Div4Q are the best strategies and their performances are not statistically different from each other.

6.6 Conclusion

In this chapter, we focused on designing new AL strategies with the goal of increasing not only the accuracy of the recommender, but also its beyond-accuracy qualities. Our approach to designing new strategies exploited an investigation into the relationships between some characteristics of the users' profiles with the performances of the recommender, which revealed that users with a profile made of diverse and unpopular items receive better recommendations. To evaluate a new strategy, we used our comprehensive evaluation framework

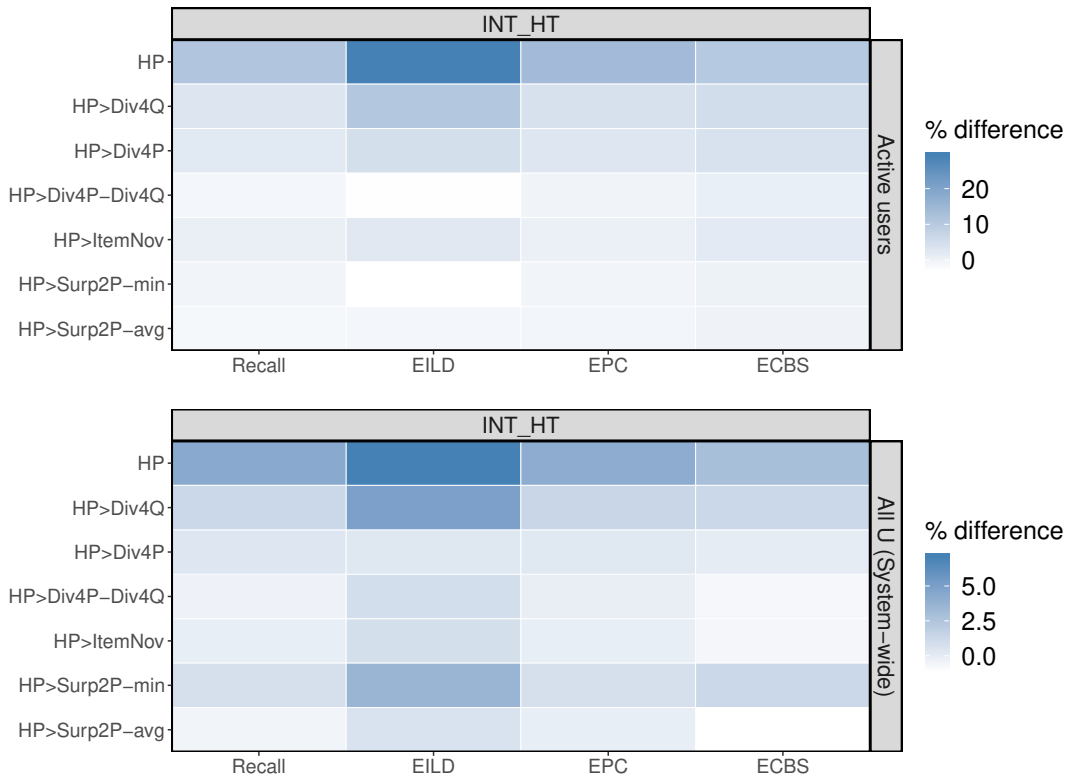

Figure 6.13: LT. The heat maps for LT show results of the cascade hybrid strategies on two groups of users, i.e. Active Users and U.

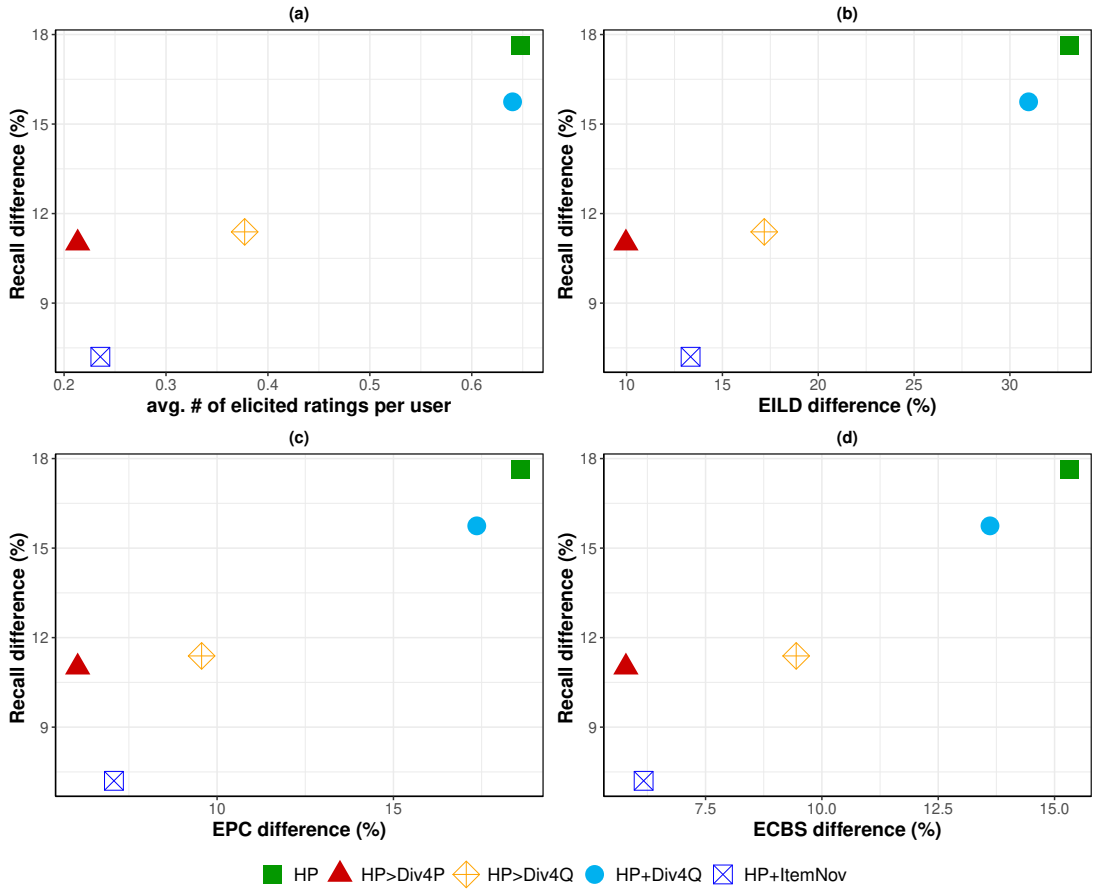

Figure 6.14: LT. Respondent users results for the best-performing hybrid strategies.

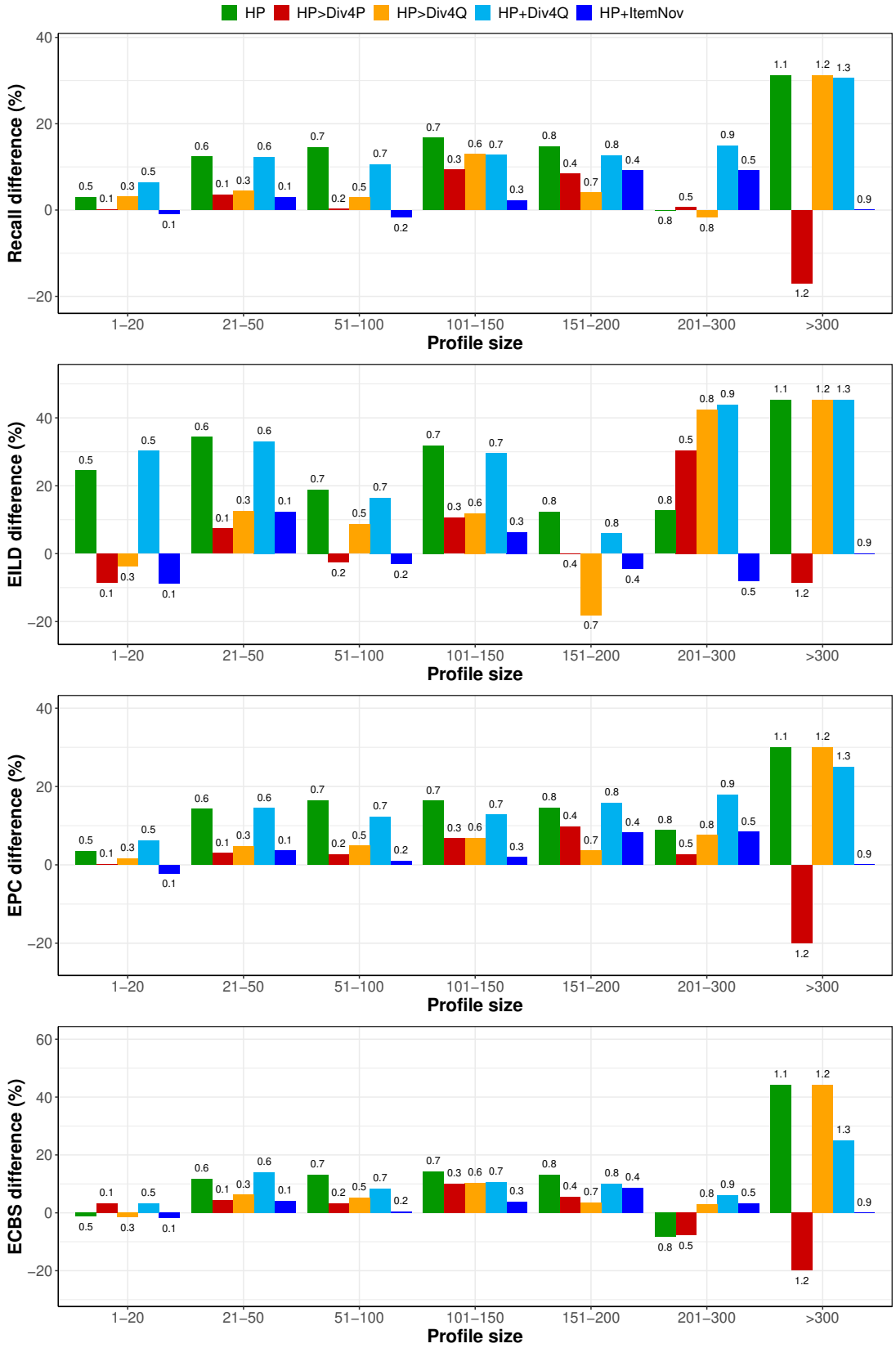

Figure 6.15: LT. Active users results by profile size, best-performing hybrid strategies.

and the datasets presented in Chapter 5.

We designed some new strategies targeting diversity, novelty and serendipity. We found that they were not only poor at increasing the specific quality that each of them is targeting, but also poor at increasing all the other metrics.

Since HP is a good strategy for eliciting lots of useful ratings, we then designed new hybrid strategies that combine the strength of HP with the strengths of some of our new proposed beyond-accuracy strategies. The new hybrid strategies improve the recommender's performance but HP still remains the best strategy at doing so (with some hybrid strategies being comparable to HP in some results). Besides HP, the weighted hybridization approach seems to be superior to the cascade in ML1M; in LT, weighted and cascade approaches do not clearly outperform each other. For both ML1M and LT, the same findings are verified when analysing results on Active users grouped based on their profile size.

In the next chapter, we draw conclusions from the work of this dissertation, and we discuss future lines of research.

Chapter 7

Conclusions & Future Work

The contributions of this dissertation focus on Active Learning and Recommender Systems, especially their offline evaluation. In particular, through the various chapters, we investigated the bias problem in RS evaluation; we addressed the question of how to best evaluate an AL strategy; and we explored new ways of designing AL strategies.

In this chapter, Section 7.1 summarizes our work and the findings of this dissertation; Section 7.2 discusses ideas for the future research.

7.1 Conclusions

Debiasing the RS evaluation

In Chapter 2, after giving an overview of the offline evaluation of RSs, we surveyed solutions proposed in the literature for the bias problem, which affects such an evaluation.

In Chapter 3, we proposed our solution to debias an offline evaluation: we described WTD and its variant WTD_H, two intervention methods that generate an unbiased-like dataset from biased data. Through an extensive set of experiments on two datasets, we found that WTD and WTD_H successfully mitigate the bias in the test data that we used, and therefore allow one to perform a debiased evaluation of an RS. In particular, WTD and WTD_H are more robust across various recommenders compared to SKEW (the existing closest intervention method to our methods), since they most closely approximate the unbiased ground-truth performances of such recommenders.

WTD and WTD_H bring several additional intrinsic benefits. They do not place overheads on the evaluation, and they are general enough to be used both with implicit and explicit datasets and to debias the training set.

A comprehensive evaluation framework for AL

In Chapter 4, we review the goals and components of Active Learning, and we give an insightful comparison of its application in classical machine learning and recommender systems.

In Chapter 5 we presented a new framework for evaluating AL for RSs, which much extends the traditional one described in the literature. It allows one to evaluate an AL strategy by considering its impact under different perspectives, e.g. on mature users and cold-start users; on users who provide new data, who do not provide any new data, and all the users in the system; and on beyond-accuracy qualities of the recommendations provided to users. Also, our framework includes two ways of mitigating the bias problem in the evaluation: one method that debiases only the test set (*INT_T*) and one that debiases both the hidden and the test sets (*INT_HT*).

We built up a case study where we assess the value of our new evaluation framework. Specifically, we compared five simple AL strategies from the literature on two widely-used biased datasets, using both the traditional methodology and our new evaluation methodology. In the experiments, we used WTD_H to debias the hidden and the test sets for our methodology. Results on both datasets show different outcomes for the two different evaluations. In particular, according to our debiased methodology, the performances of POP (which asks the users to rate the most popular items in the system) and RND (which asks the users to rate random items) are very similar, while on the classic method, largely incorrectly, POP looks better than RND. Both methods also found HP (which asks users to rate items that the recommender thinks they will like) to be the best strategy. Whether to debias both the hidden set and test set (*INT_HT*) or just the test set (*INT_T*) remains unclear, or perhaps insignificant, their results being similar.

Designing AL beyond-accuracy

In Chapter 6, we developed AL strategies that take a beyond-accuracy perspective. We argued that, because AL strategies proposed in the literature are fo-

cused on improving only recommendation accuracy, their effectiveness might be limited. In light of this, we designed some new strategies targeting diversity, novelty and serendipity, as well as accuracy. To evaluate them comprehensively, we used our *INT_HT* methodology. Experiments on two widely-used datasets show that our new strategies are successful at improving recommendation quality across the broader range of metrics, but the experiments also showed the existing HP strategy to be the best at doing so.

7.2 Future Work

In this dissertation, besides the work on the RSs offline evaluation, we shed light on Active Learning in Recommender Systems, from the offline evaluation of existing strategies to the design of new strategies. However, our research is open to further extensions. In this section, we give a brief overview of some of these opportunities.

Collecting new unbiased-like datasets

In Section 2.3 we described how to collect unbiased-like datasets by using the forced ratings approach. We also highlighted that those datasets are usually small and that this collection approach can only work in specific domains. Despite our work on debiasing data and other works in the literature too, we argue there is still the need for more unbiased data to experiment with training and evaluation of RSs. When evaluating an RS, bigger unbiased datasets would give a more grounded reference of unbiased performance. When training an RS, comparisons could be performed between models built on biased data and models built on unbiased data. For these reasons, alternatives to the forced ratings approach that are applicable across more domains and that generate bigger unbiased datasets might be investigated. Additionally, similar approaches to collecting unbiased implicit datasets might also be useful.

Debiasing the training of RSs using WTD and WTD_H

As highlighted in Section 3.6, WTD and WTD_H, our proposed methods for debiasing the offline evaluation of an RS, can also be used to debias the training set of an RS, without requiring modifications to the RS learning algorithm. Therefore, an interesting future investigation might be one that compares the

performance of an RS model built on a biased training set with a model built on a training set debiased by means of WTD or WTD_H.

Debiasing an AL strategy

In this dissertation, when evaluating AL strategies, we looked at unbiased and beyond-accuracy perspectives together; when designing new AL strategies, we looked at the beyond-accuracy perspective only. Thus, there is an opportunity to complete our research and design new strategies that try to elicit unbiased data, as well as being beyond-accuracy oriented. These would be strategies that try to elicit unbiased data from users, i.e. new data that when injected into the recommender system's training set will not increase the existing amount of bias of the training set. Works such as [FGR21] and [CDM+19] are examples that tackle the problem in machine learning scenarios.

Adaptive Active Learning

The survey of Elahi et al. [ERR16] distinguishes between personalised and non-personalised Active Learning. A strategy of the former type might request different users to rate different items, but using the same algorithm to select them; a strategy of the latter type requests all the users to rate the same items. We argue there might be room to introduce a further category. We could imagine adaptive strategies that are better targeted to the needs of different kinds of users by choosing between lower-level strategies. For example, an adaptive strategy might choose between a strategy that helps cold-start users rapidly improve recommendation relevance, but that places an emphasis on discovery for more mature users.

Verify our work in online experiments

The validity of the work in this dissertation is demonstrated with offline experiments. However, it is well-known that online experiments, such as A/B tests and user trials, are essential to give authentic insights into what has been investigated offline. This is especially true for Active Learning, where users are required to interact more deeply with the system, and the simulations that we have performed offline can only partially capture some of the signals of such interactions. Therefore, we argue that our studies should be completed with online experiments. For example, it would be interesting to build a case study

like the one we presented in Section 5.3 but with online experiments, and see how close results would be with respect to our offline results.

The preceding sections are only a sample of the main next avenues for future work. Many more suggest themselves. For example, we could investigate other ways of calculating the weights for WTD. An alternative might be using techniques developed for causal inference, e.g. [AS15, CMRR08, CMM10]. For Active Learning, we might also model its cost when designing and evaluating strategies; or propose a multi-iteration evaluation, similarly to [ERR16]; or explore Active Learning for implicit datasets. What is clear is that the topics explored in this dissertation are the starting point of much research to come.

Acronyms

AL Active Learning. xvi

BP Binary-Predicted. 77

CBS Content-Based Surprise. 13

COAT CoatShopping. 31

Div4P Diversity-For-Profile. 105

Div4PR Diversity-For-Profile-Relevant. 106

Div4Q Diversity-For-Query. 106

ECBS Expected Content-Based Surprise. 13

EILD Expected Intra-List Distance. 12

EPC Expected Popularity Complement. 12

FeatNov-av Features-Novelty-avg. 109

FeatNov-max Features-Novelty-max. 109

HP Highest-Predicted. 77

ILD Intra-List Distance. 11

ItemNov Item-Novelty. 109

KL Kullback-Leibler. 36

LT LibraryThing. 75

MAE Mean Absolute Error. 10

MAR Missing At Random. 15

MCAR Missing Completely At Random. 15

ML Machine Learning. 42

ML1M Movielens 1M. 75

MNAR Missing Not At Random. 15

NDCG Normalized Discounted Cumulative Gain. 10

PC Popularity Complement. 12

POP Popularity. 76

RMSE Root Mean Square Error. 10

RND Random. 76

RS Recommender System. xvi

S2P Similarity-To-Profile. 76

Surp2P-avg Surprise-To-Profile-avg. 111

Surp2P-min Surprise-To-Profile-min. 111

WBR3 Webscope R3. 31

References

[AAAE+15] Michal Aharon, Oren Anava, Noa Avigdor-Elgrabli, Dana Drachsler-Cohen, Shahar Golan, and Oren Somekh. Excuseme: Asking users to help in item cold-start recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, page 83–90, New York, NY, USA, 2015. Association for Computing Machinery.

[ABBC17] Marharyta Aleksandrova, Armelle Brun, Anne Boyer, and Oleg Chertov. Identifying representative users in matrix factorization-based recommender systems: application to solving the content-less new item cold-start problem. *Journal of Intelligent Information Systems*, 48(2):365–397, 2017.

[ABM17] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*, RecSys '17, page 42–46, New York, NY, USA, 2017. Association for Computing Machinery.

[ABM19] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Managing popularity bias in recommender systems with personalized re-ranking. In *The 32nd International Florida Artificial Intelligence Research Sciety (FLAIRS) Conference*, 2019.

[ABSZ14a] Gediminas Adomavicius, Jesse C Bockstedt, Curley Shawn, and Jingjing Zhang. De-biasing user preference ratings in recommender systems. In *Proceedings of the CEUR Workshop*, volume 1253, pages 2–9. CEUR-WS, 2014.

[ABSZ14b] Gediminas Adomavicius, Jesse C Bockstedt, Curley Shawn, and Jingjing Zhang. De-biasing user preference ratings in recom-

mender systems. In *Proceedings of the Workshop on Interfaces and Human Decision Making for Recommender Systems, IntRS 2014, Co-located with the 11th ACM Conference on Recommender Systems*, volume 1253 of *RecSys '17*, pages 2–9, 2014.

[ACL+90] Les Atlas, David Cohn, Richard Ladner, M. A. El-Sharkawi, and R. J. Marks. *Training Connectionist Networks with Queries and Selective Sampling*, page 566–573. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[AK12] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. on Knowl. and Data Eng.*, 24(5):896–911, May 2012.

[AKK07] Gediminas Adomavicius, Sreeharsha Kamireddy, and Youngok Kwon. Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance. In *Proceedings of the 17th Workshop on Information Technology and Systems*, 2007.

[And94] Pek van Andel. Anatomy of the Unsought Finding. Serendipity: Orgin, History, Domains, Traditions, Appearances, Patterns and Programmability. *The British Journal for the Philosophy of Science*, 45(2):631–648, 06 1994.

[ANR09] Shilpa Arora, Eric Nyberg, and Carolyn P. Rosé. Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of the NAACL Workshop on Active Learning for Natural Language Processing*, HLT '09, page 18–26, USA, 2009. Association for Computational Linguistics.

[AS15] Peter C. Austin and Elizabeth A. Stuart. Moving towards best practice when using inverse probability of treatment weighting (IPTW) using the propensity score to estimate causal treatment effects in observational studies. *Statistics in Medicine*, 34(28):3661–3679, 2015.

[AsTD09] Paul André, m.c. schraefel, Jaime Teevan, and Susan T. Dumais. Discovery is never by chance: Designing for (un)serendipity. In *Proceedings of the 7th ACM Conference on Creativity and Cognition*,

C&C '09, page 305–314, New York, NY, USA, 2009. Association for Computing Machinery.

[AT14a] Panagiotis Adamopoulos and Alexander Tuzhilin. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4), December 2014.

[AT14b] Panagiotis Adamopoulos and Alexander Tuzhilin. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Trans. Intell. Syst. Technol.*, 5(4), December 2014.

[BCC17] Alejandro Bellogín, Pablo Castells, and Iván Cantador. Statistical biases in information retrieval metrics for recommender systems. *Information Retrieval Journal*, 20(6):606–634, Dec 2017.

[BGOZ18] Jesus Bobadilla, A. Gutirrez, F. Ortega, and B. Zhu. Reliability quality measures for recommender systems. *Inf. Sci.*, 442(C):145–157, May 2018.

[BO04] Jason Baldridge and Miles Osborne. Active learning and the total cost of annotation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9–16. ACL, 2004.

[BR15] Laura Blédaité and Francesco Ricci. Pairwise preferences elicitation and exploitation for conversational collaborative filtering. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, HT '15, page 231–236, New York, NY, USA, 2015. Association for Computing Machinery.

[BR17] Matthias Braunhofer and Francesco Ricci. Selective contextual information acquisition in travel recommender systems. *Information Technology & Tourism*, 17(1):5–29, 2017.

[BRS15] Sam Banks, Rachael Rafter, and Barry Smyth. The recommendation game: Using a game-with-a-purpose to generate recommendation data. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, page 305–308, New York, NY, USA, 2015. Association for Computing Machinery.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction (UMUAI)*, 12(4):331–370, 2002.

[BV18] Stephen Bonner and Flavian Vasile. Causal embeddings for recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 104–112, New York, NY, USA, 2018. ACM.

[BVS09] Michael Bloodgood and K. Vijay-Shanker. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, CoNLL '09, page 39–47, USA, 2009. Association for Computational Linguistics.

[BZM03] Craig Boutilier, Richard S. Zemel, and Benjamin M. Marlin. Active collaborative filtering. In Christopher Meek and Uffe Kjærulff, editors, *Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, Acapulco, Mexico, August 7-10 2003*, UAI '03, pages 98–106. Morgan Kaufmann, 2003.

[CAL94] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Mach. Learn.*, 15(2):201–221, May 1994.

[CB18] Diego Carraro and Derek Bridge. A More Comprehensive Offline Evaluation of Active Learning in Recommender Systems. In *Proceedings of the Workshop on Offline Evaluation for Recommender Systems (Workshop Programme of the 12th ACM Conference on Recommender Systems)*, RecSys '18, 2018.

[CB19] Diego Carraro and Derek Bridge. Debiased offline evaluation of recommender systems: A weighted-sampling approach. In *Proceedings of REVEAL 2019, the Workshop on Reinforcement and Robust Estimators for Recommendation (Workshop Programme of the 13th ACM Conference on Recommender Systems)*, RecSys '19, 2019.

[CB20a] Diego Carraro and Derek Bridge. Debiased offline evaluation of active learning in recommender systems. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society (FLAIRS) Conference*, pages 489–494. AAAI, 2020.

[CB20b] Diego Carraro and Derek Bridge. Debiased offline evaluation of recommender systems: A weighted-sampling approach. In *Proceedings of the 35th ACM/SIGAPP Symposium on Applied Computing*, SAC '20, 2020.

[CC14] Rocío Cañamares and Pablo Castells. Exploring social network effects on popularity biases in recommender systems. In *Proceedings of the 6th Workshop on Recommender Systems and the Social Web (RSWeb 2014), collocated with the ACM RecSys 2014*, 2014.

[CC18] Pablo Castells and Rocío Cañamares. Characterization of fair experiments for recommender system evaluation–a formal analysis. In *Workshop on Offline Evaluation for Recommender Systems (REVEAL 2018) at the 12th ACM Conference on Recommender Systems (RecSys 2018). Vancouver, Canada*, 2018.

[CDM+19] Corinna Cortes, Giulia Desalvo, Mehryar Mohri, Ningshan Zhang, and Claudio Gentile. Active learning with disagreement graphs. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1379–1387. PMLR, 09–15 Jun 2019.

[CdVR08] Maarten Clements, Arjen P. de Vries, and Marcel J. T. Reinders. Optimizing single term queries using a personalized Markov random walk over the social graph. In *Proceedings of the Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 18–24, 2008.

[Cel08] Ò. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[CG98] Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA, 1998. Association for Computing Machinery.

[CGT12] Paolo Cremonesi, Franca Garzottto, and Roberto Turrin. User effort vs. accuracy in rating-based elicitation. In *Proceedings of the 6th*

ACM Conference on Recommender Systems, RecSys '12, page 27–34, New York, NY, USA, 2012. Association for Computing Machinery.

[CHT15] Shuo Chang, F. Maxwell Harper, and Loren Terveen. Using groups of items for preference elicitation in recommender systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, page 1258–1269, New York, NY, USA, 2015. Association for Computing Machinery.

[CHV15] Pablo Castells, Neil J. Hurley, and Saul Vargas. Novelty and diversity in recommender systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 881–918. Springer US, Boston, MA, 2015.

[CKMV06] Aron Culotta, Trausti Kristjansson, Andrew McCallum, and Paul Viola. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence*, 170(14):1101 – 1122, 2006.

[CKT10] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[CLA+03] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. Is seeing believing? how recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, page 585–592, New York, NY, USA, 2003. Association for Computing Machinery.

[CM05a] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence*, volume 2 of *AAAI '05*, page 746–751. AAAI Press, 2005.

[CM05b] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *Proceedings of the 20th National Conference on Artificial Intelligence*, AAAI'05, page 746–751. AAAI Press, 2005.

[CMM10] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Ad-*

vances in Neural Information Processing Systems 23, pages 442–450. Curran Associates, Inc., 2010.

[CMRR08] Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. Sample selection bias correction theory. In Yoav Freund, László Györfi, György Turán, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, pages 38–53, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[CnC17] Rocío Cañamares and Pablo Castells. A probabilistic reformulation of memory-based collaborative filtering: Implications on popularity biases. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 215–224, New York, NY, USA, 2017. Association for Computing Machinery.

[CnC18] Rocío Cañamares and Pablo Castells. Should I follow the crowd?: A probabilistic analysis of the effectiveness of popularity in recommender systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 415–424, New York, NY, USA, 2018. ACM.

[Coc77] William G. Cochran. *Sampling Techniques, 3rd Edition*. John Wiley, 1977.

[CPLS20] Andres Carvallo, Denis Parra, Hans Lobel, and Alvaro Soto. Automatic document screening of medical literature using word and text embeddings in an active learning setting. *Scientometrics*, 125(3):3047–3084, 2020.

[CR00] S. F. Chen and R. Rosenfeld. A survey of smoothing techniques for ME models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, 2000.

[CRH16] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 815–824, New York, NY, USA, 2016. Association for Computing Machinery.

[CSE18] Allison J. B. Chaney, Brandon M. Stewart, and Barbara E. Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 224–232, New York, NY, USA, 2018. ACM.

[CSP03] Giuseppe Carenini, Jocelyin Smith, and David Poole. Towards more conversational and collaborative recommender systems. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, IUI '03, pages 12–18, New York, NY, USA, 2003. ACM.

[CTP+20] Konstantina Christakopoulou, Madeleine Traverse, Trevor Potter, Emma Marriott, Daniel Li, Chris Haulk, Ed H. Chi, and Minmin Chen. Deconfounding user satisfaction estimation from response rate bias. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 450–455, New York, NY, USA, 2020. Association for Computing Machinery.

[DC08] Pinar Donmez and Jaime G. Carbonell. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 619–628, New York, NY, USA, 2008. Association for Computing Machinery.

[DCB07] Pinar Donmez, Jaime G. Carbonell, and Paul N. Bennett. Dual strategy active learning. In *Proceedings of the 18th European Conference on Machine Learning*, ECML '07, page 116–127, Berlin, Heidelberg, 2007. Springer-Verlag.

[DL10] J. Du and C. X. Ling. Active learning with human-like noisy oracle. In *Proceedings of the IEEE International Conference on Data Mining*, ICDM '10, pages 797–802, 2010.

[DSM09] Gregory Druck, Burr Settles, and Andrew McCallum. Active learning by labeling features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, volume 1 of *EMNLP '09*, page 81–90, USA, 2009. Association for Computational Linguistics.

[EBRT13] Mehdi Elahi, Matthias Braunhofer, Francesco Ricci, and Marko Tkalcic. Personality-based active learning for collaborative filtering

recommender systems. In Matteo Baldoni, Cristina Baroglio, Guido Boella, and Roberto Micalizio, editors, *Proceedings of Congress of the Italian Association for Artificial Intelligence (AI*IA 2013)*, pages 360–371, Cham, 2013. Springer International Publishing.

[Eme13] Peter Emerson. The original Borda count and partial voting. *Social Choice and Welfare*, 40(2):353–358, February 2013.

[ERR11] Mehdi Elahi, Valdemaras Repsys, and Francesco Ricci. Rating elicitation strategies for collaborative filtering. In Christian Huemer and Thomas Setzer, editors, *E-Commerce and Web Technologies*, pages 160–171, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[ERR14] Mehdi Elahi, Francesco Ricci, and Neil Rubens. Active learning strategies for rating elicitation in collaborative filtering: A system-wide perspective. *ACM Transactions on Intelligent Systems and Technology*, 5(1), January 2014.

[ERR16] Mehdi Elahi, Francesco Ricci, and Neil Rubens. A Survey of Active Learning in Collaborative Filtering Recommender Systems. *Computer Science Review*, 20(C):29–50, 2016.

[ESF18] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *Proceedings of the 41st International ACM SIGIR Conference on Research &; Development in Information Retrieval*, SIGIR '18, page 515–524, New York, NY, USA, 2018. Association for Computing Machinery.

[FGR21] Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On statistical bias in active learning: How and when to fix it. In *International Conference on Learning Representations*, 2021.

[FRH+12] Paul Felt, Eric Ringger, Kristian Heal, Robbie Haertel, and Deryle Lonsdale. First results in a study evaluating pre-annotation and correction propagation for machineassisted syriac morphological analysis. In *In Proceedings of the International Conference on Language Resources and Evaluation (LREC*, 2012.

[FTBE+16] Ignacio Fernández-Tobías, Matthias Braunhofer, Mehdi Elahi, Francesco Ricci, and Iván Cantador. Alleviating the new user problem in collaborative filtering by exploiting personality information.

User Modeling and User-Adapted Interaction, 26(2):221–255, Jun 2016.

[FTIT98] Atsushi Fujii, Takenobu Tokunaga, Kentaro Inui, and Hozumi Tanaka. Selective sampling for example-based word sense disambiguation. *Comput. Linguist.*, 24(4):573–597, December 1998.

[GG13] Ravi Ganti and Alexander G. Gray. Building bridges: Viewing active learning from the multi-armed bandit lens. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*, UAI'13, page 232–241, Arlington, Virginia, USA, 2013. AUAI Press.

[GKL10] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. On bootstrapping recommender systems. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, page 1805–1808, New York, NY, USA, 2010. Association for Computing Machinery.

[GKL11] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. Adaptive bootstrapping of recommender systems using decision trees. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, WSDM '11, page 595–604, New York, NY, USA, 2011. Association for Computing Machinery.

[Goo04] Joshua Goodman. Exponential priors for maximum entropy models. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '04, pages 305–312, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics.

[Har07] Mark A. Hart. The Long Tail: Why the Future of Business Is Selling Less of More by Chris Anderson. *Journal of Product Innovation Management*, 24(3):274–276, May 2007.

[HBOT13] Antonio Hernando, JesúS Bobadilla, Fernando Ortega, and Jorge Tejedor. Incorporating reliability measurements into the predictions of a recommender system. *Inf. Sci.*, 218:1–16, January 2013.

[HJZ10] Sheng-Jun Huang, Rong Jin, and Zhi-Hua Zhou. Active learning by querying informative and representative examples. In *Proceedings of the 23rd International Conference on Neural Information Process-*

ing Systems - Volume 1, NIPS'10, page 892–900, Red Hook, NY, USA, 2010. Curran Associates Inc.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions Information Systems (TOIS)*, 22(1):5–53, January 2004.

[HLHG14] José Miguel Hernández-Lobato, Neil Houlsby, and Zoubin Ghahramani. Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, volume 32 of *ICML'14*, pages II–1512–II–1520. JMLR.org, 2014.

[Hof04] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, January 2004.

[HRC08] Robbie A. Haertel, Eric K. Ringger, and James L. Carroll. Return on investment for active learning, 2008.

[HRJL08] S. C. H. Hoi, Rong Jin, Jianke Zhu, and M. R. Lyu. Semi-supervised svm batch mode active learning for image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–7, 2008.

[HSRF95] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, page 194–201, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[Hwa04] Rebecca Hwa. Sample selection for statistical parsing. *Comput. Linguist.*, 30(3):253–276, September 2004.

[HY08] Abhay S. Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 91–98, New York, NY, USA, 2008. ACM.

[IR15] Guido W. Imbens and Donald B. Rubin. *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge University Press, 2015.

[JS04] Rong Jin and Luo Si. A bayesian approach toward active learning for collaborative filtering. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, page 278–285, Arlington, Virginia, USA, 2004. AUAI Press.

[JS16] Thorsten Joachims and Adith Swaminathan. Sigir tutorial on counterfactual evaluation and learning for search, recommendation and ad placement. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1199–1201. ACM, July 2016.

[JSS17] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, WSDM '17, page 781–789, New York, NY, USA, 2017. Association for Computing Machinery.

[JVG18] Olivier Jeunen, Koen Verstrepen, and Bart Goethals. Fair offline evaluation methodologies for implicit-feedback recommender systems with mnar data. In *Proceedings of the 1st International Workshop on Offline Evaluation for Recommender Systems (REVEAL), colocated with the 12th ACM Conference on Recommender Systems*, RecSys '18, 2018.

[KB14] Marius Kaminskas and Derek Bridge. Measuring surprise in recommender systems. In Panagiotis Adamopoulos, Alejandro BellogÃn, Pablo Castells, Paolo Cremonesi, and Harald Steck, editors, *Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design (Workshop Programme of the 8th ACM Conference on Recommender Systems)*, RecSys '14, 2014.

[KB17] Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 7(1):2:1–2:42, March 2017.

[KBV09] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[KC14] Yong-Deok Kim and Seungjin Choi. Bayesian binomial mixture model for collaborative prediction with non-random missing data. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 201–208, New York, NY, USA, 2014. ACM.

[Ken38] M. G. Kendall. A New Measure or Rank Correlation. *Biometrika*, 30(1-2):81–93, 06 1938.

[KFNS11] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Non-myopic active learning for recommender systems based on matrix factorization. In *Proceedings of the IEEE International Conference on Information Reuse Integration (IRI)*, pages 299–303, 2011.

[KHB07] Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 877–882, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[KK14] Daniel Kluver and Joseph A. Konstan. Evaluating recommender behavior for new users. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 121–128, New York, NY, USA, 2014. ACM.

[KKZV18] Denis Kotkov, Joseph A. Konstan, Qian Zhao, and Jari Veijalainen. Investigating serendipity in recommender systems based on real user feedback. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1341–1350, New York, NY, USA, 2018. Association for Computing Machinery.

[KM01] Arnd Kohrs and Bernard Mérialdo. Improving collaborative filtering for new-users by smart object selection. In *Proceedings of the International Conference on Media Futures (ICME)*, Florence, ITALY, 05 2001.

[KNST14] Rasoul Karimi, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Improved questionnaire trees for active learning in recommender

systems. In *Proceedings of the Workshop on Lernen, Wissen, Adaption (LWA)*, 2014.

[KNST15] Rasoul Karimi, Alexandros Nanopoulos, and Lars Schmidt-Thieme. A supervised active learning framework for recommender systems based on decision trees. *User Modeling and User-Adapted Interaction*, 25(1):39–64, Mar 2015.

[Kor09] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 447–456, New York, NY, USA, 2009. Association for Computing Machinery.

[Kor10] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1), January 2010.

[KR20] Walid Krichene and Steffen Rendle. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1748–1757, New York, NY, USA, 2020. Association for Computing Machinery.

[KRG18] Saikishore Kalloori, Francesco Ricci, and Rosella Gennari. Eliciting pairwise preferences in recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 329–337, New York, NY, USA, 2018. Association for Computing Machinery.

[KS11] Yehuda Koren and Joe Sill. Ordrec: An ordinal model for predicting personalized item rating distributions. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 117–124, New York, NY, USA, 2011. Association for Computing Machinery.

[KWJ+04] Ross D King, Kenneth E Whelan, Ffion M Jones, Philip G K Reiser, Christopher H Bryant, Stephen H Muggleton, Douglas B Kell, and Stephen G Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247—252, January 2004.

[Lat10] Neal Lathia. *Evaluating Collaborative Filtering Over Time*. PhD thesis, University College London, UK, 2010.

[LB92] Kenneth Lang and Eric Baum. Query learning can work poorly when a human oracle is used. In *Proceedings of the International Joint Conference on Neural Network (IJCNN)*, 1992.

[LBA+07] R. Lomasky, C. E. Brodley, M. Aernecke, D. Walt, and M. Friedl. Active class selection. In Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, editors, *Proceedings of the 18th European Conference on Machine Learning*, pages 640–647, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[LC94] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *Machine Learning Proceedings 1994*, pages 148 – 156. Morgan Kaufmann, San Francisco (CA), 1994.

[LCB16] Dawen Liang, Laurent Charlin, and David M. Blei. Causal inference for recommendation. In *Proceedings of the Workshop on Causation, co-located with the Uncertainty in Artificial Intelligence (UAI) Conference*, 2016.

[LCMB16] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 951–961, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[LEKS+18] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9725–9735. Curran Associates, Inc., 2018.

[LG94] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th International ACM SIGIR Conference on Research and Development in Informa-*

tion Retrieval, SIGIR '94, page 3–12, Berlin, Heidelberg, 1994. Springer-Verlag.

[LHZ14] Benedikt Loepp, Tim Hussein, and Jüergen Ziegler. Choice-based preference elicitation for collaborative filtering recommender systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 3085–3094, New York, NY, USA, 2014. Association for Computing Machinery.

[LML15] Daryl Lim, Julian McAuley, and Gert Lanckriet. Top-n recommendation with missing implicit feedback. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 309–312, New York, NY, USA, 2015. ACM.

[LMLY11] Nathan N. Liu, Xiangrui Meng, Chao Liu, and Qiang Yang. Wisdom of the better few: Cold start recommendation via representative based rating elicitation. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, page 37–44, New York, NY, USA, 2011. Association for Computing Machinery.

[LR86] Roderick J A Little and Donald B Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[MAZ10] Carlos Eduardo Mello, Marie-Aude Aufaure, and Geraldo Zimbrao. Active learning driven by rating impact analysis. In *Proceedings of the 4th ACM Conference on Recommender Systems*, RecSys '10, page 341–344, New York, NY, USA, 2010. Association for Computing Machinery.

[Maz13] Maciej A. Mazurowski. Estimating confidence of individual rating predictions in collaborative filtering recommender systems. *Expert Syst. Appl.*, 40(10):3847–3857, August 2013.

[ME19] Farshad Bakhshandegan Moghaddam and Mehdi Elahi. Cold start solutions for recommendation systems. *Big Data Recommender Systems - Volume 2: Application Paradigms*, pages 35–56, 2019.

[MGGR14] Arnaud De Myttenaere, Bénédicte Le Grand, Boris Golden, and Fabrice Rossi. Reducing offline evaluation bias in recommendation systems. *Proceedings of BENELEARN Conference*, 2014.

[MJ92] Robert R. McCrae and Oliver P. John. An Introduction to the Five-Factor Model and Its Applications. *Journal of Personality*, 60(2):175–215, June 1992.

[MLG+03] Sean Mcnee, Shyong K. Lam, Catherine Guetzlaff, Joseph A. Konstan, and John Riedl. Confidence displays and training in recommender systems. In *Proceedings of the 9th IFIP TC13 International Conference on HumanComputer Interaction (INTERACT*, pages 176–183. IOS Press, 2003.

[MN98] Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 350–358, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[MNS+07a] Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher, Roman Englert, and Yuval Elovici. Improving the detection of unknown computer worms activity using active learning. In *Proceedings of the 30th Annual German Conference on Advances in Artificial Intelligence*, KI '07, page 489–493, Berlin, Heidelberg, 2007. Springer-Verlag.

[MNS+07b] Robert Moskovitch, Nir Nissim, Dima Stopel, Clint Feher, Roman Englert, and Yuval Elovici. Improving the detection of unknown computer worms activity using active learning. In *Proceedings of the 30th Annual German Conference on Advances in Artificial Intelligence*, KI '07, page 489–493, Berlin, Heidelberg, 2007. Springer-Verlag.

[MRK06] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *Proceedings of the CHI Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, page 1097–1101, New York, NY, USA, 2006. Association for Computing Machinery.

[MZ09] Benjamin M. Marlin and Richard S. Zemel. Collaborative prediction and ranking with non-random missing data. In *Third ACM Conference on Recommender Systems (RecSys '09)*, pages 5–12, 2009.

[MZRS07] Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. In *Twenty-Third Conference on Uncertainty in Artificial Intelligence (UAI'07)*, pages 267–275, 2007.

[MZS20] Alessandro B. Melchiorre, Eva Zangerle, and Markus Schedl. Personality bias of music recommendation algorithms. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 533–538, New York, NY, USA, 2020. Association for Computing Machinery.

[NS04] Hieu T. Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine Learning*, ICML '04, page 79, New York, NY, USA, 2004. Association for Computing Machinery.

[OPY+11] J. Oh, S. Park, H. Yu, M. Song, and S. Park. Novel recommendation based on personal popularity tendency. In *Proceedings of the IEEE 11th International Conference on Data Mining (ICDM)*, pages 507–516, 2011.

[OT09] Fredrik Olsson and Katrin Tomanek. An intrinsic stopping criterion for committee-based active learning. In *Proceedings of the 13th Conference on Computational Natural Language Learning*, CoNLL '09, page 138–146, USA, 2009. Association for Computational Linguistics.

[PJN12] Xuan Hau Pham, Jason J. Jung, and Ngoc-Thanh Nguyen. Integrating multiple experts for correction process in interactive recommendation systems. In *Proceedings of the 4th International Conference on Computational Collective Intelligence: Technologies and Applications - Volume Part I*, ICCCI'12, page 31–40, Berlin, Heidelberg, 2012. Springer-Verlag.

[PQEC17] Roberto Pagano, Massimo Quadrana, Mehdi Elahi, and Paolo Cremonesi. Toward active learning in cross-domain recommender systems. *CoRR*, abs/1701.02021, 2017.

[PUG12] Bruno Pradel, Nicolas Usunier, and Patrick Gallinari. Ranking with non-random missing ratings: Influence of popularity and positivity on evaluation metrics. In *Proceedings of the 6th ACM Conference on*

Recommender Systems, RecSys '12, pages 147–154, New York, NY, USA, 2012. ACM.

[PZT10] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. Fast ALS-based matrix factorization for explicit and implicit feedback datasets. In *Proceedings of the 4th ACM Conference on Recommender Systems*, RecSys '10, pages 71–78, New York, NY, USA, 2010. ACM.

[RAC+02] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, page 127–134, New York, NY, USA, 2002. Association for Computing Machinery.

[RKR08] Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: An information theoretic approach. *SIGKDD Explorations Newsletters*, 10(2):90–100, December 2008.

[RM01] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, page 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[RMJ06] Hema Raghavan, Omid Madani, and Rosie Jones. Active learning with feedback on features and instances. *J. Mach. Learn. Res.*, 7:1655–1686, December 2006.

[RS07] Neil Rubens and Masashi Sugiyama. Influence-based collaborative active learning. In *Proceedings of the 1st ACM Conference on Recommender Systems*, RecSys '07, page 145–148, New York, NY, USA, 2007. Association for Computing Machinery.

[SB20] Tobias Schnabel and Paul N. Bennett. Debiasing item-to-item recommendations with small annotated datasets. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 73–81, New York, NY, USA, 2020. Association for Computing Machinery.

[SC08] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, page 1070–1079, USA, 2008. Association for Computational Linguistics.

[SCF08] Burr Settles, Mark W. Craven, and Lewis A. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, 2008.

[SCR07] Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 1289–1296, Red Hook, NY, USA, 2007. Curran Associates Inc.

[SDW01] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden Markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, IDA '01, page 309–318, Berlin, Heidelberg, 2001. Springer-Verlag.

[Set11] Burr Settles. Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, page 1467–1478, USA, 2011. Association for Computational Linguistics.

[Set12] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, Boston, MA, 2011.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[SJ15] Adith Swaminathan and Thorsten Joachims. Counterfactual risk minimization: Learning from logged bandit feedback. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International*

Conference on Machine Learning, volume 37 of *ICML '15*, pages 814–823, Lille, France, 07–09 Jul 2015. PMLR.

[SKRdR18] Anna Sepliarskaia, Julia Kiseleva, Filip Radlinski, and Maarten de Rijke. Preference elicitation as an optimization problem. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 172–180, New York, NY, USA, 2018. Association for Computing Machinery.

[SLH10] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*, RecSys '10, page 269–272, New York, NY, USA, 2010. Association for Computing Machinery.

[SM01] Barry Smyth and Paul McClave. Similarity vs. diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, ICCBR '01, page 347–361, Berlin, Heidelberg, 2001. Springer-Verlag.

[SOS92] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, COLT '92, page 287–294, New York, NY, USA, 1992. Association for Computing Machinery.

[SRA20] Tobias Schnabel, Gonzalo Ramos, and Saleema Amershi. "Who Doesn't like Dinosaurs?" Finding and Eliciting Richer Preferences for Recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 398–407, New York, NY, USA, 2020. Association for Computing Machinery.

[SSS+16] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *Proceedings of the 33rd International Conference on Machine Learning*, New York, NY, USA, 2016.

[Ste10] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 713–722, 2010.

[Ste11] Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, pages 125–132, New York, NY, USA, 2011. ACM.

[Ste13] Harald Steck. Evaluation of recommendations: Rating-prediction and ranking. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 213–220, New York, NY, USA, 2013. ACM.

[SU07] Andrew I. Schein and Lyle H. Ungar. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265, August 2007.

[SW11] Shaun Seaman and Ian White. Review of inverse probability weighting for dealing with missing data. *Statistical methods in medical research*, 22, 05 2011.

[TCRC02] Ivan R. Teixeira, Francisco de A. T. de Carvalho, Geber Ramalho, and Vincent Corruble. ActiveCP: A method for speeding up user preferences acquisition in collaborative filtering systems. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, SBIA '02, page 237–247, Berlin, Heidelberg, 2002. Springer-Verlag.

[Tho12] Steven K. Thompson. *Sampling*. John Wiley & Sons, Ltd, 2012.

[VBKC14] Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, and Pablo Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 209–216, New York, NY, USA, 2014. Association for Computing Machinery.

[VC11] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, page 109–116, New York, NY, USA, 2011. Association for Computing Machinery.

[VG09] S. Vijayanarasimhan and K. Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2262–2269, 2009.

[VHE+18] Kristen Vaccaro, Dylan Huang, Motahhare Eslami, Christian Sandvig, Kevin Hamilton, and Karrie Karahalios. The illusion of control: Placebo effects of control settings. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '18, page 1–13, New York, NY, USA, 2018. Association for Computing Machinery.

[VJG10] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-sighted active learning on a budget for image and video recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3035–3042, 2010.

[Vla08] Andreas Vlachos. A stopping criterion for active learning. *Comput. Speech Lang.*, 22(3):295–312, July 2008.

[VYP17] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. Dropoutnet: Addressing cold start in recommender systems. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4957–4966. Curran Associates, Inc., 2017.

[WLCB18] Yixin Wang, Dawen Liang, Laurent Charlin, and David M. Blei. The deconfounded recommender: A causal inference approach to recommendation. *CoRR*, abs/1808.06581, 2018.

[WMZ10] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28(4), November 2010.

[WZ09] Jun Wang and Jianhan Zhu. Portfolio theory of information retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 115–122, New York, NY, USA, 2009. Association for Computing Machinery.

[XYT+03] Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. Representative sampling for text classification using support vector machines. In *Proceedings of the 25th European Conference on IR Research*, ECIR'03, page 393–407, Berlin, Heidelberg, 2003. Springer-Verlag.

[XYT+15] Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, and Jian Lu. Ice-breaking: Mitigating cold-start recommendation problem

by rating comparison. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2015-January, pages 3981–3987. International Joint Conferences on Artificial Intelligence, 2015. 24th International Joint Conference on Artificial Intelligence, IJCAI 2015 ; Conference date: 25-07-2015 Through 31-07-2015.

[YAR08] Emine Yilmaz, Javed A. Aslam, and Stephen Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, page 587–594, New York, NY, USA, 2008. Association for Computing Machinery.

[YCX+18] Longqi Yang, Yin Cui, Yuan Xuan, Chenyang Wang, Serge Belongie, and Deborah Estrin. Unbiased offline recommender evaluation for missing-not-at-random implicit feedback. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 279–287, New York, NY, USA, 2018. ACM.

[YLH+20] Bowen Yuan, Yaxu Liu, Jui-Yang Hsia, Zhenhua Dong, and Chih-Jen Lin. Unbiased ad click prediction for position-aware advertising systems. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, page 368–377, New York, NY, USA, 2020. Association for Computing Machinery.

[YRFD11] Yan Yan, Romer Rosales, Glenn Fung, and Jennifer G. Dy. Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning*, ICML'11, page 1161–1168, Madison, WI, USA, 2011. Omnipress.

[Yu05] Hwanjo Yu. Svm selective sampling for ranking with application to data retrieval. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, page 354–363, New York, NY, USA, 2005. Association for Computing Machinery.

[ZC15] Chicheng Zhang and Kamalika Chaudhuri. Active learning from weak and strong labelers. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 703–711, Cambridge, MA, USA, 2015. MIT Press.

[Zha13] Liang Zhang. The Definition of Novelty in Recommendation System. *Journal of Engineering Science and Technology Review*, 6(3):141–145, June 2013.

[ZKL+10] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

[ZLG03] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the ICML Workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining (Co-located with KDD '03)*, pages 58–65, 2003.

[ZLH+20] Y. Zhu, J. Lin, S. He, B. Wang, Z. Guan, H. Liu, and D. Cai. Addressing the item cold-start problem by attribute-driven active learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(4):631–644, 2020.

[ZMKL05] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 22–32, New York, NY, USA, 2005. Association for Computing Machinery.

[ZOBG18] B. Zhu, F. Ortega, J. Bobadilla, and A. Gutiérrez. Assigning reliability values to recommendations using matrix factorization. *Journal of Computational Science*, 26:165 – 177, 2018.

[ZYZ11] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 315–324, New York, NY, USA, 2011. Association for Computing Machinery.

[ZZL17] Xiaoying Zhang, Junzhou Zhao, and John C.S. Lui. Modeling the assimilation-contrast effects in online product rating systems: Debiasing and recommendations. In *Proceedings of the 11th ACM Conference on Recommender Systems*, RecSys '17, page 98–106, New York, NY, USA, 2017. Association for Computing Machinery.

Appendix A

Statistical significance tests

The tables in this appendix report the statistical significance results for the experiments performed in Chapters 3, 5 and 6. In each table, we place the value 'TRUE' where a statistical significant difference in the performance is found. For such tests, we use a two-tailed Wilcoxon signed rank test with $p < 0.05$. The version of the test that we use includes zero-differences in the ranking process and assigns a zero rank to those differences. Also, it splits the zero ranks between positive and negative ones.

A.1 Results of Chapter 3

Table A.1: The statistical significance of the results is assessed by performing a pairwise comparison test between the performance of each recommender on the five different test sets, i.e. the baselines sets FULL, REG and the intervened sets SKEW, WTD and WTD_H.

	WBR3					COAT				
	FULL	REG	SKEW	WTD	WTD_H	FULL	REG	SKEW	WTD	WTD_H
	PosPop									
FULL	-	-	-	-	-	-	-	-	-	-
REG		-	-	-	-		-	-	-	-
SKEW	TRUE	TRUE	-	-	-	TRUE	TRUE	-	-	-
WTD	TRUE	TRUE	TRUE	-	-	TRUE	TRUE		-	-
WTD_H	TRUE	TRUE	TRUE	TRUE	-	TRUE	TRUE			-
	AvgRating									
FULL	-	-	-	-	-	-	-	-	-	-
REG		-	-	-	-		-	-	-	-
SKEW	TRUE	TRUE	-	-	-	TRUE		-	-	-
WTD	TRUE	TRUE		-	-	TRUE			-	-
WTD_H	TRUE	TRUE	TRUE		-	TRUE				-
	UB_KNN									
FULL	-	-	-	-	-	-	-	-	-	-
REG	TRUE	-	-	-	-		-	-	-	-
SKEW	TRUE	TRUE	-	-	-	TRUE	TRUE	-	-	-
WTD	TRUE	TRUE	TRUE	-	-	TRUE	TRUE	TRUE	-	-
WTD_H	TRUE	TRUE	TRUE	TRUE	-	TRUE	TRUE	TRUE		-
	IB_KNN									
FULL	-	-	-	-	-	-	-	-	-	-
REG	TRUE	-	-	-	-	TRUE	-	-	-	-
SKEW	TRUE	TRUE	-	-	-	TRUE	TRUE	-	-	-
WTD	TRUE	TRUE	TRUE	-	-	TRUE	TRUE	TRUE	-	-
WTD_H	TRUE	TRUE	TRUE	TRUE	-	TRUE	TRUE	TRUE		-
	MF									
FULL	-	-	-	-	-	-	-	-	-	-
REG	TRUE	-	-	-	-		-	-	-	-
SKEW	TRUE	TRUE	-	-	-	TRUE		-	-	-
WTD	TRUE	TRUE	TRUE	-	-	TRUE	TRUE	TRUE	-	-
WTD_H	TRUE	TRUE	TRUE	TRUE	-	TRUE	TRUE	TRUE		-

Table A.2: Statistical significance results for WBR3. We perform a pairwise comparison test between the performances of the recommenders on the unbiased test set D^{gt}.

	PosPop	AvgRating	UB_KNN	IB_KNN	MF
PosPop	-	-	-	-	-
AvgRating	TRUE	-	-	-	-
UB_KNN	TRUE	TRUE	-	-	-
IB_KNN	TRUE	TRUE		-	-
MF	TRUE	TRUE	TRUE	TRUE	-

Table A.3: Statistical significance results for COAT. We perform a pairwise comparison test between the performances of the recommenders on the unbiased test set D^{gt}.

	PosPop	AvgRating	UB_KNN	IB_KNN	MF
PosPop	-	-	-	-	-
AvgRating		-	-	-	-
UB_KNN			-	-	-
IB_KNN				-	-
MF				TRUE	-

A.2 Results of Chapter 5

Table A.4: The statistical significance of the results of ML1M for Active users is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	CLASSIC						INT_T						INT_HT							
	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP		
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	Recall	
POP	TRUE	-	-		TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-			TRUE	-	-	-				
HP	TRUE	-	-	-	-		TRUE	-	-	-	-		TRUE	-	-	-	-			
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	EILD	
POP	TRUE	-	-		TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-			TRUE	-	-	-				
HP	TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-		TRUE	-	-	-	-			
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	EPC	
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-			TRUE	-	-	-				
HP	TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE		
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		
RND		-	TRUE	TRUE	TRUE	TRUE		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	ECBS	
POP	TRUE	-	-	TRUE		TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-			TRUE	-	-	-				
HP	TRUE	-	-	-	-		TRUE	-	-	-	-		TRUE	-	-	-	-			
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		

Table A.5: The statistical significance of the results of ML1M for the system-wide perspective is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

		CLASSIC					INT_T					INT_HT					
	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP	
RND		-	TRUE	TRUE	TRUE	TRUE	TRUE	-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	Recall
POP	TRUE	-	-		TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-			TRUE	-	-	-			
HP	TRUE	-	-	-	-		TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	EILD
POP	TRUE	-	-		TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE		TRUE	-	-	-			TRUE	-	-	-			
HP	TRUE	-	-	-	-		TRUE	-	-	-	-		TRUE	-	-	-	-		
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	EPC
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE		TRUE	-	-	-			TRUE	-	-	-			
HP	TRUE	-	-	-	-		TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	
RND		-	TRUE	TRUE	TRUE	TRUE	TRUE	-		TRUE	TRUE	TRUE		-		TRUE	TRUE		ECBS
POP	TRUE	-	-					-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE		TRUE	-	-	-			TRUE	-	-	-			
HP	TRUE	-	-	-	-		TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		-	-	-	-	-	

Table A.6: The statistical significance of the results of ML1M for the Active users grouped in buckets is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

		Recall				EILD				EPC				ECBS				
		After AL iteration	S2P	HP	BP	After AL iteration	S2P	HP	BP	After AL iteration	S2P	HP	BP	After AL iteration	S2P	HP	BP	
S2P		TRUE	-				-				-				-			1-20
HP		TRUE		-				-		TRUE		-		TRUE		-		
BP		TRUE		TRUE	-				-	TRUE		TRUE	-				-	
S2P		TRUE	-				-			TRUE	-				-			21-50
HP		TRUE		-				-		TRUE		-		TRUE		-		
BP		TRUE			-				-	TRUE			-	TRUE		TRUE	-	
S2P		TRUE	-				-			TRUE	-			TRUE	-			51-100
HP		TRUE		-				-		TRUE		-		TRUE		-		
BP		TRUE			-				-	TRUE		TRUE	-	TRUE		TRUE	-	
S2P		TRUE	-			TRUE	-			TRUE	-				-			101-150
HP		TRUE		-				-		TRUE		-		TRUE		-		
BP		TRUE			-	TRUE			-	TRUE			-			TRUE	-	
S2P			-				-			TRUE	-				-			151-200
HP				-				-		TRUE		-				-		
BP		TRUE		TRUE	-				-	TRUE			-				-	
S2P			-				-				-				-			201-300
HP				-				-				-				-		
BP					-				-				-				-	
S2P			-				-				-				-			300
HP				-				-				-				-		
BP					-				-				-				-	

Table A.7: The statistical significance of the results of LT for the Active users is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

		CLASSIC					INT_T					INT_HT					
	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP	
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	Recall
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-	TRUE		TRUE	-	-	-	TRUE		
HP	TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-		
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	
RND		-	TRUE	TRUE	TRUE	TRUE		-	TRUE	TRUE	TRUE	TRUE		-	TRUE	TRUE	TRUE	TRUE	EILD
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-	TRUE	TRUE	
HP	TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-	TRUE	
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	EPC
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-	TRUE		TRUE	-	-	-	TRUE		
HP	TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-	TRUE	
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	ECBS
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE	
S2P	TRUE	-	-	-	TRUE		TRUE	-	-	-	-		TRUE	-	-	-	-		
HP	TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-	TRUE	
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	

Table A.8: The statistical significance of the results of LT for the system-wide perspective is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	CLASSIC						INT_T						INT_HT							
	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP	After AL iteration	RND	POP	S2P	HP	BP		
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	Recall	
POP		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P		-	-	-	TRUE		TRUE	-	-	-			TRUE	-	-	-				
HP		-	-	-	-		TRUE	-	-	-	-		TRUE	-	-	-	-			
BP		-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	EILD	
POP		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-			TRUE	-	-	-				
HP	TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-	TRUE		
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE	EPC	
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P	TRUE	-	-	-	TRUE	TRUE	TRUE	-	-	-			TRUE	-	-	-				
HP	TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-			
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		
RND		-	TRUE	TRUE	TRUE	TRUE		-		TRUE	TRUE	TRUE		-		TRUE	TRUE		ECBS	
POP	TRUE	-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		-	-	TRUE	TRUE	TRUE		
S2P	TRUE	-	-	-	TRUE		TRUE	-	-	-			TRUE	-	-	-				
HP	TRUE	-	-	-	-		TRUE	-	-	-	-	TRUE	TRUE	-	-	-	-			
BP	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-	TRUE	-	-	-	-	-		

Table A.9: The statistical significance of the results of LT for the Active users grouped in buckets is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after an AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	Recall				EILD				EPC				ECBS				
	After AL iteration	S2P	HP	BP	After AL iteration	S2P	HP	BP	After AL iteration	S2P	HP	BP	After AL iteration	S2P	HP	BP	
S2P		-				-				-				-			1-20
HP			-				-				-				-		
BP				-				-				-				-	
S2P	TRUE	-			TRUE	-			TRUE	-			TRUE	-			21-50
HP	TRUE		-		TRUE		-		TRUE		-		TRUE		-		
BP	TRUE			-	TRUE			-	TRUE			-	TRUE			-	
S2P	TRUE	-	TRUE	TRUE	TRUE	-			TRUE	-	TRUE	TRUE	TRUE	-		TRUE	51-100
HP	TRUE		-		TRUE		-		TRUE		-		TRUE		-		
BP	TRUE			-	TRUE			-	TRUE			-	TRUE			-	
S2P		-				-				-				-			101-150
HP	TRUE		-		TRUE		-		TRUE		-		TRUE		-		
BP				-	TRUE			-	TRUE			-				-	
S2P		-				-				-				-			151-200
HP			-				-				-				-		
BP				-				-				-				-	
S2P		-				-				-				-			201-300
HP			-				-				-				-		
BP				-				-				-				-	
S2P		-				-				-				-			>300
HP			-				-				-				-		
BP				-				-				-				-	

A.3 Results of Chapter 6

Table A.10: ML1M. The statistical significance of the results of the weighted hybrid strategies for the Active users is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP+Div4P	HP+Div4Q	HP+Div4P-Div4Q	HP+ItemNov	HP+Surp2P-min	HP+Surp2P-avg	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Recall
HP+Div4P	TRUE	-	-	TRUE	TRUE				
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-	TRUE		TRUE	
HP+ItemNov	TRUE	-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	EILD
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE		TRUE		
HP+Div4P-Div4Q		-	-	-	-	TRUE		TRUE	
HP+ItemNov		-	-	-	-	-	-		
HP+Surp2P-min		-	-	-	-	-	-	-	
HP+Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	EPC
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-	TRUE		TRUE	
HP+ItemNov	TRUE	-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg	TRUE	-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	ECBS
HP+Div4P	TRUE	-	-	-	-				
HP+Div4Q	TRUE	-	-	-	TRUE		TRUE		
HP+Div4P-Div4Q		-	-	-	-	-			
HP+ItemNov	TRUE	-	-	-	-	-	-		
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	

Table A.11: ML1M. The statistical significance of the results of the weighted hybrid strategies for the System-wide perspective is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP+Div4P	HP+Div4Q	HP+Div4P-Div4Q	HP+ItemNov	HP+Surp2P-min	HP+Surp2P-avg	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P	TRUE	-	-	TRUE	TRUE		TRUE		
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-			TRUE	Recall
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg	TRUE	-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE		TRUE		
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE		TRUE		
HP+Div4P-Div4Q		-	-	-	-				EILD
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg	TRUE	-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P	TRUE	-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-			TRUE	EPC
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg	TRUE	-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE		TRUE	
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE		
HP+Div4P-Div4Q		-	-	-	-				ECBS
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg	TRUE	-	-	-	-	-	-	-	

Table A.12: ML1M. The statistical significance of the results of the cascade hybrid strategies for the Active users is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP>Div4P	HP>Div4Q	HP>Div4P-Div4Q	HP>ItemNov	HP>Surp2P-min	HP>Surp2P-avg	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Recall
HP>Div4P		-	-	TRUE			TRUE		
HP>Div4Q	TRUE	-	-	-	TRUE	TRUE		TRUE	
HP>Div4P-Div4Q		-	-	-	-		TRUE		
HP>ItemNov		-	-	-	-	-	TRUE		
HP>Surp2P-min	TRUE	-	-	-	-	-	-	TRUE	
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	EILD
HP>Div4P		-	-	TRUE					
HP>Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP>Div4P-Div4Q		-	-	-	-	-			
HP>ItemNov	TRUE	-	-	-	-	-	-	-	
HP>Surp2P-min		-	-	-	-	-	-	-	
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	EPC
HP>Div4P		-	-	TRUE					
HP>Div4Q	TRUE	-	-	-	TRUE	TRUE		TRUE	
HP>Div4P-Div4Q		-	-	-	-		TRUE		
HP>ItemNov		-	-	-	-	-			
HP>Surp2P-min	TRUE	-	-	-	-	-	-	TRUE	
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	ECBS
HP>Div4P		-	-						
HP>Div4Q	TRUE	-	-	-	TRUE	TRUE		TRUE	
HP>Div4P-Div4Q		-	-	-	-		TRUE		
HP>ItemNov		-	-	-	-	-			
HP>Surp2P-min	TRUE	-	-	-	-	-	-	TRUE	
HP>Surp2P-avg		-	-	-	-	-	-	-	

Table A.13: ML1M. The statistical significance of the results of the cascade hybrid strategies for the System-wide perspective is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP>Div4P	HP>Div4Q	HP>Div4P-Div4Q	HP>ItemNov	HP>Surp2P-min	HP>Surp2P-avg	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Recall
HP>Div4P	-	-	-						
HP>Div4Q	TRUE	-	-	-	TRUE	TRUE		TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-	TRUE		
HP>Surp2P-min	TRUE	-	-	-	-	-	-	TRUE	
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	EILD
HP>Div4P	-	-	-				TRUE		
HP>Div4Q	TRUE	-	-	-		TRUE		TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-	TRUE		
HP>Surp2P-min		-	-	-	-	-	-	-	
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	EPC
HP>Div4P	-	-	-						
HP>Div4Q	TRUE	-	-	-	TRUE	TRUE		TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-	TRUE		
HP>Surp2P-min		-	-	-	-	-	-	TRUE	
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE		TRUE	ECBS
HP>Div4P	-	-	-						
HP>Div4Q	TRUE	-	-	-	TRUE	TRUE		TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-			
HP>Surp2P-min		-	-	-	-	-	-	TRUE	
HP>Surp2P-avg		-	-	-	-	-	-	-	

Table A.14: LT. The statistical significance of the results of the weighted hybrid strategies for the Active users is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP+Div4P	HP+Div4Q	HP+Div4P-Div4Q	HP+ItemNov	HP+Surp2P-min	HP+Surp2P-avg	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P		-	-	TRUE			TRUE		
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-				Recall
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P		-	-	TRUE		TRUE			
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-	TRUE			EILD
HP+ItemNov		-	-	-	-	-		TRUE	
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-				EPC
HP+ItemNov		-	-	-	-	-	TRUE		
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-				ECBS
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	

Table A.15: LT. The statistical significance of the results of the weighted hybrid strategies for the System-wide perspective is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP+Div4P	HP+Div4Q	HP+Div4P-Div4Q	HP+ItemNov	HP+Surp2P-min	HP+Surp2P-avg	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P	TRUE	-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-				Recall
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg	TRUE	-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P		-	-						
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-				EILD
HP+ItemNov		-	-	-	-	-		TRUE	
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q		-	-	-	-				EPC
HP+ItemNov		-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE		TRUE	TRUE	TRUE	TRUE	
HP+Div4P		-	-	TRUE					
HP+Div4Q	TRUE	-	-	-	TRUE	TRUE	TRUE	TRUE	
HP+Div4P-Div4Q	TRUE	-	-	-	-				ECBS
HP+ItemNov	TRUE	-	-	-	-	-			
HP+Surp2P-min		-	-	-	-	-	-		
HP+Surp2P-avg		-	-	-	-	-	-	-	

Table A.16: LT. The statistical significance of the results of the cascade hybrid strategies for the Active users is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP>Div4P	HP>Div4Q	HP>Div4P-Div4Q	HP>ItemNov	HP>Surp2P-min	HP>Surp2P-avg	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Recall
HP>Div4P		-	-						
HP>Div4Q	TRUE	-	-	-	TRUE		TRUE	TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-			
HP>Surp2P-min		-	-	-	-	-	-		
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	EILD
HP>Div4P		-	-						
HP>Div4Q	TRUE	-	-	-	TRUE		TRUE	TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-	TRUE		
HP>Surp2P-min		-	-	-	-	-	-		
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	EPC
HP>Div4P	TRUE	-	-		TRUE		TRUE		
HP>Div4Q	TRUE	-	-	-	TRUE		TRUE	TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-			
HP>Surp2P-min		-	-	-	-	-	-		
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	ECBS
HP>Div4P	TRUE	-	-						
HP>Div4Q	TRUE	-	-	-			TRUE	TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov		-	-	-	-	-			
HP>Surp2P-min		-	-	-	-	-	-		
HP>Surp2P-avg		-	-	-	-	-	-	-	

Table A.17: LT. The statistical significance of the results of the cascade hybrid strategies for the System-wide perspective is assessed by performing two sets of tests. The first applies to each AL strategy independently, where we determine if the recommender performance before the AL iteration is statistically different from its performance after the AL strategy has elicited some ratings (i.e. after the AL iteration). The second set of tests are pairwise comparison tests between AL strategies, to determine if the recommender's performance after the AL iteration performed by one strategy is statistically different from the recommender's performance after the AL iteration performed by another strategy.

	After AL iteration	HP	HP>Div4P	HP>Div4Q	HP>Div4P-Div4Q	HP>ItemNov	HP>Surp2P-min	HP>Surp2P-avg	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	Recall
HP>Div4P	TRUE	-	-				TRUE		
HP>Div4Q	TRUE	-	-	-	TRUE		TRUE	TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov	TRUE	-	-	-	-	-			
HP>Surp2P-min		-	-		-	-	-		
HP>Surp2P-avg		-	-		-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	EILD
HP>Div4P		-	-						
HP>Div4Q	TRUE	-	-	-	TRUE		TRUE		
HP>Div4P-Div4Q		-	-	-	-	-	TRUE		
HP>ItemNov		-	-	-	-	-	-		
HP>Surp2P-min		-	-	-	-	-	-	-	
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	EPC
HP>Div4P	TRUE	-	-		TRUE		TRUE		
HP>Div4Q	TRUE	-	-	-	TRUE		TRUE	TRUE	
HP>Div4P-Div4Q		-	-	-	-				
HP>ItemNov	TRUE	-	-	-	-	-			
HP>Surp2P-min		-	-	-	-	-	-		
HP>Surp2P-avg		-	-	-	-	-	-	-	
HP	TRUE	-			TRUE		TRUE	TRUE	ECBS
HP>Div4P	TRUE	-	-		TRUE				
HP>Div4Q	TRUE	-	-	-	TRUE		TRUE	TRUE	
HP>Div4P-Div4Q	TRUE	-	-	-	-				
HP>ItemNov	TRUE	-	-	-	-	-			
HP>Surp2P-min	TRUE	-	-	-	-	-	-		
HP>Surp2P-avg		-	-	-	-	-	-	-	

Table A.18: ML1M. We report the statistical significance of the results for the Active users grouped in buckets based on their profile-size. 'T' stands for a significant result. We rename the strategies as 'B' for HP+Div4P, 'C' for HP>Div4Q, 'D' for HP+Div4Q, 'E' for HP+ItemNov.

	Recall						EILD						EPC						ECBS						
	After AL iteration	HP	B	C	D	E	After AL iteration	HP	B	C	D	E	After AL iteration	HP	B	C	D	E	After AL iteration	HP	B	C	D	E	
HP	T	-		T		T		-					T	-		T		T	T	-	T	T		T	
B	T	-	-			T		-	-				T	-	-					-	-		T		
C		-	-	-	T	T	T	-	-	-				-	-	-	T	T		-	-	-		T	1-20
D	T	-	-	-	-	T	T	-	-	-	-		T	-	-	-	-	T	T	-	-	-	-	T	
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP	T	-		T		T	T	-		T		T	T	-		T		T	T	-					
B	T	-	-	T		T	T	-	-				T	-	-	T		T	T	-	-			T	
C	T	-	-	-	T	T		-	-	-	T	T	T	-	-	-	T	T	T	-	-	-			21-50
D	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP	T	-		T		T	T	-				T	T	-		T		T	T	-					
B	T	-	-			T	T	-	-			T	T	-	-			T		-	-				
C	T	-	-	-	T	T	T	-	-	-		T	T	-	-	-	T	T	T	-	-	-			51-100
D	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP	T	-						-					T	-					T	-					
B	T	-	-	T	T		T	-	-				T	-	-	T			T	-	-				
C	T	-	-	-				-	-	-			T	-	-	-				-	-	-			101-150
D		-	-	-	-			-	-	-	-		T	-	-	-	-			-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP		-	T					-					T	-						-					
B	T	-	-	T				-	-				T	-	-					-	-				
C		-	-	-				-	-	-				-	-	-				-	-	-			151-200
D		-	-	-	-			-	-	-	-			-	-	-	-		T	-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP		-						-						-						-					
B		-	-					-	-					-	-					-	-				
C		-	-	-				-	-	-				-	-	-				-	-	-			201-300
D		-	-	-	-			-	-	-	-			-	-	-	-			-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP		-						-						-						-					
B		-	-					-	-					-	-					-	-				
C		-	-	-				-	-	-				-	-	-				-	-	-			>300
D		-	-	-	-			-	-	-	-			-	-	-	-			-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	

Table A.19: LT. We report the statistical significance of the results for the Active users grouped in buckets based on their profile-size. 'T' stands for a significant result. We rename the strategies as 'B' for HP>Div4P, 'C' for HP>Div4Q, 'D' for HP+Div4Q, 'E' for HP+ItemNov.

	Recall						EILD						EPC						ECBS						
	After AL iteration	HP	B	C	D	E	After AL iteration	HP	B	C	D	E	After AL iteration	HP	B	C	D	E	After AL iteration	HP	B	C	D	E	
HP		-			T			-						-						-					
B		-	-					-	-					-	-					-	-				
C		-	-	-				-	-	-	T			-	-	-				-	-	-			1-20
D		-	-	-	-			-	-	-	-			-	-	-	-			-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP	T	-	T	T		T	T	-	T	T		T	T	-	T	T		T	T	-	T	T		T	
B	T	-	-		T			-	-		T		T	-	-		T			-	-		T		
C		-	-	-	T	T		-	-	-	T	T	T	-	-	-	T	T	T	-	-	-	T	T	21-50
D	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-	T	
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP	T	-	T	T		T	T	-	T			T	T	-	T	T	T	T	T	-	T	T		T	
B		-	-		T			-	-		T			-	-		T			-	-				
C		-	-	-		T		-	-	-		T		-	-	-	T	T		-	-	-		T	51-100
D	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-	T	T	-	-	-	-	T	
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP	T	-				T	T	-					T	-	T			T	T	-					
B		-	-					-	-					-	-					-	-				
C	T	-	-	-		T		-	-	-				-	-	-		T		-	-	-			101-150
D		-	-	-	-			-	-	-	-		T	-	-	-	-	T		-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP		-						-						-						-					
B		-	-					-	-					-	-					-	-				
C		-	-	-				-	-	-				-	-	-				-	-	-			151-200
D		-	-	-	-			-	-	-	-			-	-	-	-			-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP		-						-						-						-					
B		-	-					-	-					-	-		T			-	-				
C		-	-	-				-	-	-				-	-	-				-	-	-			201-300
D		-	-	-	-			-	-	-	-		T	-	-	-	-			-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	
HP		-						-						-						-					
B		-	-					-	-					-	-					-	-				
C		-	-	-				-	-	-				-	-	-				-	-	-			>300
D		-	-	-	-			-	-	-	-			-	-	-	-			-	-	-	-		
E		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-		-	-	-	-	-	